НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Програмна система для визначення статі автора у художніх творах на підставі аналізу згадок кольорів*

Виконав: студент IV курсу, групи *ІП-63 Попова Віра Юріївна* _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н, Фіногенов О.Д.* _____
посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада, науковий ступінь, вчене звання, прізвище, і ім'я, по батькові (підпис)

Рецензент:

доцент каф. АПЕПС, ТЕФ, к.т.н. Ольга ЗАЛЕВСЬКА
посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1004053772

Дата перевірки:
15.06.2020 16:58:07 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2020 17:08:47 EEST

ID користувача:
77149

Назва документу: Popova_ip63_2

ID файлу: 1004066670 **Кількість сторінок:** 57 **Кількість слів:** 8863 **Кількість символів:** 71958 **Розмір файлу:** 1.80 MB

4.41% Схожість

Найбільша схожість: 1.39% з джерело https://habr.com/ru/post/280238

3.67% Схожість з Інтернет джерелами	46	Page 59
2.28% Текстові збіги по Бібліотеці акаунту	85	Page 59

0% Цитат

Не знайдено жодних цитат

79.5% Вилучень

Джерела менше, ніж 8 слів автоматично вилучено

Не знайдено жодного вилученого тексту з Інтернету

79.5% Вилученого тексту з Бібліотеки	1	Page 59

Підміна символів

Не знайдено замінених символів

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Поповій Вірі Юріївні
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Програмна система для визначення статі автора у художніх творах на підставі аналізу згадок кольорів*»

керівник проєкту Олексій Дмитрович Фіногенов *к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: загальні положення, опис предметної області, огляд існуючих технічних рішень, розробка функціональних та нефункціональних вимог, постановка комплексу завдань модуля

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки даних

3) Розгортання та впровадження програмного забезпечення: опис процесів

тестування, опис контрольного прикладу

4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, робота з програмним забезпеченням, супровід програмного забезпечення

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна бізнес-процесів

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	19.03.2020	
2.	*Аналіз існуючих методів розв'язання задачі*	26.03.2020	
3.	*Постановка та формалізація задачі*	26.03.2020	
4.	*Аналіз вимог до програмного забезпечення*	02.04.2020	
5.	*Алгоритмізація задачі*	02.04.2020	
6.	*Моделювання програмного забезпечення*	09.04.2020	
7.	*Обґрунтування використовуваних технічних засобів*	16.04.2020	
8.	*Розробка архітектури програмного забезпечення*	23.04.2020	
9.	*Розробка програмного забезпечення*	30.04.2020	
10.	*Налагодження програми*	07.05.2020	
11.	*Виконання графічних документів*	*14.05.2020*	
12.	*Оформлення пояснювальної записки*	*21.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*06.06.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Віра ПОПОВА
(підпис)

Керівник _____ Олексій ФІНОГЕНОВ
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	A4		Завдання на дипломний проєкт	2	
2	A4	КПІ.ІП-6324.045490.02.81	Пояснювальна записка	63	
3	A4	КПІ.ІП-6324.045490.03.91	Технічне завдання	12	
4	A4	КПІ.ІП-6324.045490.04.51	Програма та методика тестування	6	
5	A4	КПІ.ІП-6324.045490.05.34	Керівництво користувача	5	
6	A4	КПІ.ІП-6324.045490.06.13	Опис програми	14	
7	A3	КПІ.ІП-6324.045490.07.99.СС	Схема структурна варіантів використання	1	
8	A3	КПІ.ІП-6324.045490.08.99.СС	Схема структурна бізнес-процесів	1	
9	A3	КПІ.ІП-6324.045490.09.99.КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6324.045490.01.90		
Розробн.		Попова В.Ю.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Фіногенов О.Д					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 17 таблиць, 14 рисунків та 13 джерел – загалом 63 сторінки. Об`єкт дослідження: художні твори українських авторів.

Мета дипломного проекту: можливість визначення статі автора тексту художніх творів на підставі аналізу згадок кольорів. У першому розділі наведено аналіз предметної області, аналіз відомих технічних рішень та аналіз потенційних конкурентів. На основі дослідження сформовано вимоги до проекту. Другий розділ присвячений моделюванню та конструюванню програмного забезпечення, розробці архітектури, бази даних. Третій розділ містить опис аналізу якості програми та загальний опис етапів та процесів тестування. У четвертому розділі описано інструменту щодо розгортання та супроводу даного програмного забезпечення. Також наведено: опис програми, інструкцію користувача, програму та методику тестування, креслення вигляду екранних форм, схему структурну варіантів використань, схему структурну бізнес-процесів.

КЛЮЧОВІ СЛОВА: АНАЛІЗ, АТРИБУЦІЯ, АЛГОРИТМ, ПРОГРАМНА СИСТЕМА, КЛАСИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 17 tables, 14 figures and 14 sources - a total of 63 pages. Object of research: works of art by Ukrainian authors.

The purpose of the diploma project: the ability to determine the sex of the author of the text of works of art based on the analysis of color mentions. The first section provides an analysis of the subject area, analysis of known technical solutions and analysis of potential competitors. Based on the study, the requirements for the project are formed. The second section is devoted to modeling and design of software, development of architecture, database. The third section contains a description of the program quality analysis and a general description of the stages and processes of testing. The fourth section describes the tools for deploying and maintaining this software. Also provided: described programs, testing methods, software and testing methods, development and implementation of forms, structure of structural options, structure of structural schemes.

KEY WORDS: ANALYSIS, ATTRIBUTION, ALGORITHM, SOFTWARE SYSTEM, CLASSIFICATION, NEURAL NETWORK.

Пояснювальна записка
до дипломного проєкту

на тему: *Програмна система для визначення статі автора у художніх творах на підставі аналізу згадок кольорів*

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP - процес обробки природних мов (англ. Natural Language Processing), галузь штучного інтелекту, яка займається взаємодією між комп'ютерами та людьми за допомогою природної мови.

RGB – (англ. Red, Green, Blue) кольорова модель, яка визначає спосіб кодування кольору за допомоги трьох кольорів.

Стемінг - процес, який позбавляє слова закінчень, намагаючись зробити це і часто передбачає видалення похідних афіксів.

Парсинг – процес читання тексту та перетворення його у більш корисний формат.

ВСТУП

У сучасному світі кожну годину з’являється величезна кількість нової інформації. Кожна людина щоденно стикається з нескінченним потоком новин, статей та іншого виду інформацією. Наразі технології досягли такого розвитку, що час, за який інформація проходить від етапу створення до етапу споживання користувачем, займає лічені хвилини. Велику частку цієї інформації становлять авторські тексти. Саме у цей час постає завдання визначення авторства невідомого тексту, характеристик тощо. Авторство є міждисциплінарною галуззю, яка спрямована на розробку логіки ідентифікації авторства невідомого тексту. Від самого початку це завдання вирішувалось спеціалістами у даній галузі. Спираючись на свій досвід, знання, а іноді інтуїцію, вони висували гіпотези, які згодом намагались довести, здійснюючи аналіз літературного стилю, змісту у контексті історичних подій, періоду часу, приймаючи до уваги філософські, соціальні та інші погляди автора. Однак, потрібно врахувати, що атрибуція, здійснена людиною є виснажливою, важкою та займає багато часу. Тексти, які піддаються аналізу найчастіше мають великий обсяг. Під час такого аналізу, спеціаліст може пропустити важливі деталі, або знехтувати об’єктивною оцінкою через суб’єктивний погляд на зміст тексту.

Стрімкий розвиток технологій дав поштовх для широкого використання машинного навчання, інтелектуального аналізу даних та процесів природної мови у багатьох галузях. Були розроблені нові методи атрибуції, які використовуються у комп’ютерному аналізі. Автоматизована атрибуція дає результати, які не містять суб’єктивної оцінки та можуть бути перевірені незалежно. Цей підхід знайшов широке застосування у різних галузях, зокрема у літературі, цифрових правах, виявленні плагіату, криміналістиці, лінгвістиці та інших.

Наразі немає системи для дослідження україномовного тексту з метою визначення авторства. Спираючись на дослідження, які стверджують різницю у сприйнятті кольорів та образністю мислення між жінками та чоловіками,

створена система для аналізу україномовного твору з метою визначення авторства. Досліджуваною характеристикою у даному випадку є стать автора. Система аналізує використання автором кольорів як художнього засобу при написанні художніх творів.

Завданнями розробки є створення датасету для навчання нейронної мережі, словника, який містить інформацію про кольори, їх назви та аналогії, створення системи графіків для відображення отриманих результатів.

Створена програмна система може бути використана для завдань різних галузей, метою яких є визначення статі або дослідження кольорових характеристик тексту як засіб опису навколишнього світу, передачі суб'єктивного бачення автора.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

На сьогоднішній день існують відомі методи визначення авторства тексту. Вони включають завдання атрибуції. Атрибуція тексту являє собою процес дослідження тексту маючи за мету встановлення авторства або визначення певних характеристик належних автору і умови створення певного текстового документа.

Атрибуція тексту поділяється два завдання:

– ідентифікація;

– діагностика.

Обидва завдання вирішуються в тих умовах, що автор тексту є невідомим. Діагностичне завдання дозволяє визначити характеристики, які є притаманними певному автору тексту. Прикладом таких характеристик є освітній рівень автора, національна приналежність, історична епоха, до якої належить автор. Проблема атрибуції текстів є поширеною у безлічі галузей та є невід'ємною для дослідження філологами, лінгвістами, юристами, криміналістами, істориками та іншими. У теперішній час задля атрибуції текстів використовуються підходи з теорії розпізнання образів, математичної статистики, кластерного аналізу, алгоритмів нейронних мереж та безліч інших.

Варто зазначити, що не існує єдиного алгоритму визначення авторства невідомого тексту. Це народжує різноманітні підходи і несподівані знахідки серед дослідників даної задачі. В українській науці накопичено чимало даних, які свідчать про помітну виразність у авторських текстах характеристик, які є різноманітними індивідуальними властивостями кожного окремого автора. Серед них можна побачити статеві, вікові, соціальні і національні ознаки. Також важливим фактором є емоційні особливості та інтелектуальна сфера автора, а також індивідуально-особистісні характеристики. В українській лінгвістиці в названому напрямку робляться лише перші кроки. Поки що не

існує апробованих методів визначення характеристик автора українського тексту.

У даній роботі розглянута така визначальна характеристика як використання різних назв кольорів. Метою є можливість визначення статі автора, використовуючи цю характеристику. Припущення валідності даної характеристики робиться, виходячи з результатів досліджень, які підтверджують вимірну відмінність сприйняття і використання кольорів в залежності від статі. У своїй роботі Шапаніс, який є одним з засновників ергономіки, виявив, що жінки були значно більш послідовними у відповідності елементів кольорового спектру до кольорових назв [1]. Крім того, повідомлялося, що жінки мають більший набір слів і використовують більш детальні терміни для опису кольору. Було встановлено, що чоловіки звертають меншу вагу при розділенні уздовж червоно-зеленої осі. У нещодавньому веб-психологічному експерименті [2], у необмеженому позначанні кольорових назв, жінки виявили набагато більш широкий перелік слів, означаючих кольори, включаючи велику різноманітність неосновних кольорових термінів та "фантазійні" назви кольорів, у свою чергу коли чоловіки вживали більш основні терміни та їх сполучення. Далі було встановлено, що, порівняно з чоловіками, жінки виявили більш досконалу мовну сегментацію кольорового простору, переважно вздовж червоно-зеленої осі кольорового простору.

Ці висновки можуть відображати гендерні відмінності в культурних факторах, що стосуються діапазону доступних кольорових термінів. За минулі роки проведено велику кількість досліджень, щоб визначити, які типи фізіологічних відмінностей можуть існувати між чоловіками та жінками в різних аспектах кольорового зору. Повідомлялося про відмінності в унікальному зовнішньому вигляді, а також про різні показники експериментів із відповідності кольорів. Зазначалось, що у жінок більш широкий діапазон відповідності. У кількох інших останніх дослідженнях було зафіксовано значні відмінності у зорових функціях чоловіка та жінки, пов’язані із зовнішнім

виглядом кольору та периферичним зором. Сезель та декілька інших дослідників вивчали відображення слів, визначаючих кольори та кольорових образів між різними підгрупами спостерігачів та знайшов докази для різних уявлень про мовну схожість у різних групах [3]. Дослідження, про які повідомлялося вище, охоплюють цілий спектр галузей - від фізіології до психології та лінгвістики.

Стає очевидним, що в обробці кольорової інформації між чоловіками та жінками існують відмінності, що ці відмінності спостерігаються на різних рівнях, починаючи від сприйняття до кольорової лексики, і можуть бути віднесені до цілого ряду механізмів, таких як генетичного та фізіологічного до поведінкового. Генетичні відмінності між чоловіками та жінками приписуються генам Х-хромосоми, що кодують фото пігменти [4]. Дві статі відрізняються репертуаром частих кольорових термінів: байєвський синтетичний спостерігач виявив, що жінки поділяють кольоровий простір лінгвістично щільніше в «теплій» зоні, тоді як чоловіки роблять це в «прохолодній» зоні.

У своїй роботі, спираючись на перелічені та інші багаточисленні дослідження та праці, я розробила систему для аналізу україномовного твору з метою визначення статі автора тексту, спираючись на те, як він використовує назви кольорів у своїй роботі.

1.2 Змістовний опис і аналіз предметної області

Процес обробки природних мов (англ. Natural Language Processing) представляє собою технологію, яка направлена розпізнавання змістового сенсу природної мови людини [5]. Обробка природних мов, як правило скорочена як NLP - це галузь штучного інтелекту, яка займається взаємодією між комп'ютерами та людьми за допомогою природної мови. Кінцевою метою NLP є читати, розшифровувати, розуміти та осмислювати людські мови таким чином, щоб інформація, яка отримується на виході мала якомога найбільшу цінність. Більшість методик NLP покладаються на машинне навчання, щоб

отримати сенс з написаного людською мовою. Обробка природних мов є рушійною силою наступних загальних застосувань. Насамперед це програми для перекладу мови, такі як Google Translate. Текстові процесори, такі як Microsoft Word і Grammarly, які використовують NLP для перевірки граматичної точності текстів. Програми інтерактивної голосової відповіді (IVR), що використовуються в телефонних центрах для відповіді на запити певних користувачів. Персональні програми-помічники, такі як OK Google, Siri, Cortana та Alexa. Обробка природних мов вважається важкою проблемою в інформатиці. Природа людської мови ускладнює НЛП. Правила, якими зумовлена передача інформації за допомогою природних мов, для комп'ютерів нелегкі для розуміння. Деякі з цих правил можуть бути високорівневими та абстрактними. Це можна проілюструвати наступним прикладом. Ситуація, коли хтось використовує саркастичне зауваження для передачі інформації. Воно буде легко зрозумілим для людини та важким завданням для машини. Всебічне розуміння людської мови вимагає розуміння не тільки слів, а і того, як пов'язані поняття задля досягнення певного сенсу. Незважаючи на те, що люди можуть легко опанувати мову, неоднозначність та неточність характеристик природних мов - це те, що робить NLP важким для машинного впровадження. Обробка природних мов тягне за собою застосування алгоритмів для виявлення та вилучення природних мовних правил таким чином, щоб неструктуровані мовні дані перетворювались у форму, яку комп'ютери можуть зрозуміти. Отримавши вхідний текст, комп'ютер буде використовувати алгоритми, щоб витягувати значення, пов'язані з кожним реченням, і збирати з них суттєві дані. Синтаксичний аналіз та семантичний аналіз представляють основні прийоми, що використовуються для виконання завдань з обробки природних мов. Синтаксис позначає розташування слів у реченні таким чином, щоб вони мали граматичний зміст. При обробці природних мов синтаксичний аналіз використовується для оцінки того, як природна мова узгоджується з

граматичними правилами. Комп'ютерні алгоритми використовуються для застосування граматичних правил до групи слів та отримання значень від них.

Зазвичай використовуються такі синтаксичні прийоми:

– лематизація;
– морфологічна сегментація;
– токенізація;
– стеммінг.

Лематизація передбачає приведення слів до їх словникової форми. Канонічну форму слова називають лемою слова [6]. Щоб лематизація розв’язала слово до його леми, вона повинна знати його частину мови. Для цього потрібна додаткова обчислювальна лінгвістика, наприклад, частина мови. Це дозволяє процесу робити більш якісні рішення. Ще одна річ, яку слід зазначити про лематизацію, полягає в тому, що дуже часто створення лематизатора для нової мови є більш важким завданням, ніж створення основного алгоритму. Оскільки лематизаторам потрібно набагато більше знань про структуру мови, це набагато інтенсивніший процес, ніж налаштування евристичного алгоритму. Підсумовуючи, можна сказати, що лематизація це зведення різних складних форм слова в єдину форму для легкого аналізу.

Морфологічна сегментація передбачає поділ слів на окремі одиниці, що називаються морфемами.

Токенізація слів, яку також називають сегментацією слів, є проблемою поділу рядка тексту на її складові слова. В англійській та багатьох інших мовах, що використовують певну форму латинського алфавіту, пробіл є гарним наближенням розділювача слів. Однак все ще можуть виникнути проблеми. Деякі англійські складені іменники пишуться варіативно, а іноді містять пробіл. У більшості випадків використовують бібліотеки для досягнення бажаних результатів.

Стемінг є, безумовно, простішим за лематизацію. Під час стемінгу слова зводяться до їхніх коренів слова. Корінь слова не повинен бути таким самим

коренем, як морфологічний корінь на основі словника. Він є рівним або меншим ніж слово.

Важливим терміном в обробці природних мов є семантичний аналіз. Семантика - це значення, яке передається текстом. Семантичний аналіз - один із складних аспектів обробки природних мов, який ще не був повністю вирішений. Він передбачає застосування комп'ютерних алгоритмів для розуміння значення та тлумачення слів та того, як структуруються речення.

Розпізнавання іменованої сутності (NER) є одним з методів семантичного аналізу [7]. Воно включає визначення частин тексту, які можна ідентифікувати та класифікувати за попередньо заданими групами. Приклади таких груп включають імена людей та назви місць. Ще один метод має назву розбіжність сенсу у слові. Він передбачає надання сенсу слову на основі контексту. Генерація природних мов передбачає використання баз даних для отримання смислових намірів та перетворення їх на людську мову.

Обробка природних мов грає вирішальну роль у підтримці машинно-людських взаємодій. Оскільки в цій галузі проводиться все більше досліджень, очікується, що далі можна буде побачити більше проривів, які зроблять машини розумнішими у визнанні та розумінні людської мови.

Основною задачею в даній роботі була розробка програмної систему для визначення статі автора на підставі аналізу згадок кольору в творі. З точки зору групування текстів, все зводиться до бінарної класифікації. Класифікація являє собою завдання, яке вимагає використання алгоритмів, які у свою чергу призначають мітку класу даних на прикладах із проблемної області. Легким для розуміння прикладом є класифікація електронних листів як "спам" або "не спам". Існує багато різних типів завдань класифікації, які зустрічаються у машинному навчанні та спеціалізованих підходах до моделювання, які можуть бути використані для кожного. Класифікаційне передбачуване моделювання передбачає призначення мітки класу для введення прикладів. Бінарна класифікація відноситься до прогнозування одного з двох класів, а багато

класова класифікація передбачає передбачення одного з більш ніж двох класів. Класифікація, що містить багато міток, передбачає передбачення одного або декількох класів для кожного прикладу, а незбалансована класифікація стосується завдань класифікації, де розподіл прикладів по класах не є рівним. У даному випадку це завдання бінарної класифікації.

Існує багато різних типів алгоритмів класифікації для моделювання задач прогнозування. Можна зазначити, що не існує істинно вірної теорії щодо вибору алгоритмів для певних типів проблем. Замість цього, як правило, рекомендується користуватись контрольними експериментами та виявляти, який алгоритм або конфігурація алгоритму призводить до найкращої ефективності для даного завдання класифікації. Алгоритми прогнозування оцінюються на основі їх результатів. Точність класифікації є важливим показником, який використовується для оцінки продуктивності моделі на основі передбачуваних міток класу. Точність класифікації не є досконалою, але є хорошою відправною точкою для багатьох завдань класифікації. Замість міток класів деякі задачі можуть вимагати прогнозування ймовірності членства в класі для кожного прикладу. Це забезпечує додаткову невизначеність у прогнозі, яку програма чи користувач може потім інтерпретувати. Популярною діагностикою для прогнозованих ймовірностей є крива ROC [8].

Зазвичай завдання бінарної класифікації містить у собі один клас, який є нормальним станом, і інший клас, який є аномальним. Це можна проілюструвати прикладом, який наводився вище. У цьому випадку "не спам" являється нормальним станом, а "спам" - це аномальний стан. Інший приклад стосується медичного тесту, у якому «рак не виявлений» буде позначатись як нормальний стан, а «виявлений рак» відповідно аномальний стан. Класу для нормального стану присвоюється мітка 0, а класу з аномальним станом присвоюється мітка 1.

Дуже поширеним підходом до моделювання задач бінарної класифікації є прогнозування, яке передбачає розподіл ймовірностей Бернуллі для кожного

прикладу. Розподілом Бернуллі називають дискретний розподіл ймовірностей, який охоплює випадок, коли подія матиме двійковий результат як 0 або 1. Для класифікації це означає, що модель передбачає ймовірність прикладу, що належить до класу 1, або аномального стану.

Можна перерахувати кілька популярних алгоритмів, які використовують для задач бінарної класифікації:

- логістична регресія;
- метод k-найближчих сусідів;
- дерево рішень;
- метод опорних векторів;
- наївний Байєс;
- нейронна мережа.

Деякі алгоритми розроблені спеціально для бінарної класифікації і не підтримують більше двох класів. Далі буде викладено докладніше про кожен з алгоритмів.

Вище перелічені методи використовуються для задач бінарної класифікації. Ця робота передбачає наявність двох класів, таких як жінки та чоловіки. Хочу зазначити, що на даний момент немає системи для дослідження україномовного тесту з метою визначення авторства. Вхідними даними було обрано художні твори українських письменників. Цей вибір зумовлений декількома причинами. По-перше, були обрані саме українські автори, з метою дослідження мовних особливостей наших співвітчизників. По-друге, були обрані художні твори, насамперед тому, що вони здебільшого мають достатньо велику кількість слів для побудови відповідної моделі. Також, саме у художніх текстах автори використовують багату палітру описових прийомів, які зазвичай поширено використовують назви кольорів для яскравої передачі читачу настрою та атмосфери.

1.3 Аналіз успішних ІТ-проектів

На сьогоднішній час не існує програмного забезпечення для визначення статі автора україномовного тексту. У англомовній літературі зустрічаються дослідження та розробки з ідентифікації статі автора, які використовують методи та підходи які описані нижче. Хочу зазначити, що в них не використовується така характеристика як колір та його використання представниками різної статі.

1.3.1 Аналіз відомих технічних рішень

Логістична регресія, або логіт-регресія, є популярною статистичною моделлю, яка використовується для двійкової класифікації, тобто для прогнозів типу так чи ні, А або В [9]. На Рисунку 1.1 проілюстровано роботу логістичної моделі, де зеленим позначена границя, яка розмежовує точки на площині на два класи.

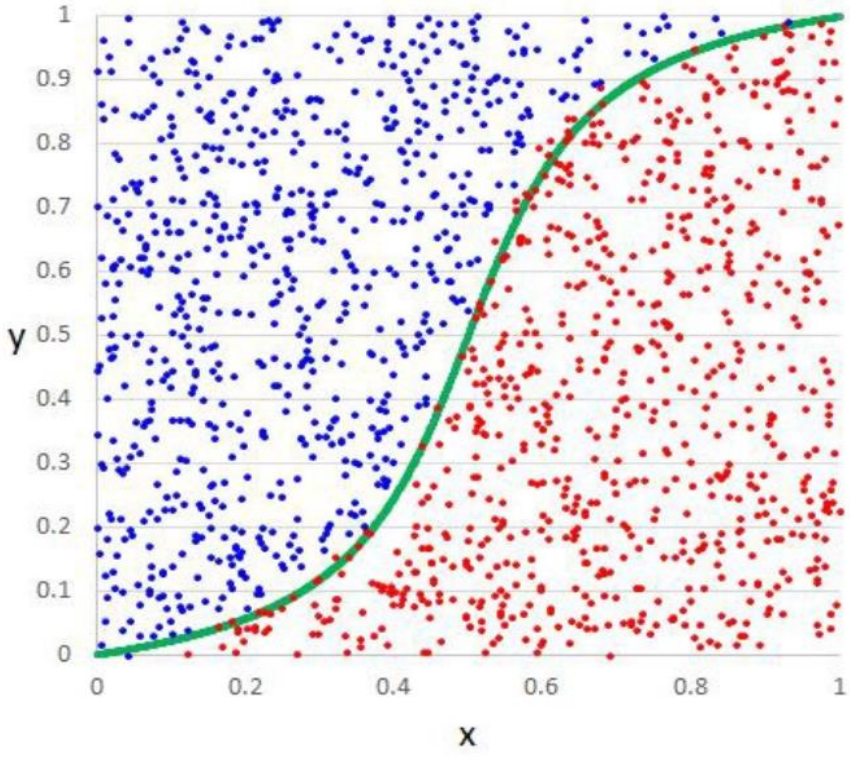


Рисунок 1.1 – Приклад логістичної регрессії

Логістична регресія, однак, може використовуватись і для багато класової класифікації. Логістична регресія є простою, але потужною лінійною моделлю, яка є формою регресії між 0 і 1 на основі вектору вхідних ознак. Визначаючи значення відсічення, яке за замовчуванням дорівнює 0,5, для класифікації використовується модель регресії. Зазвичай моделі машинного навчання базуються на деякій математичній функції, яка обробляє вхідні функції та видає значення. Цю функцію називають гіпотезою h(x). Логістична функція представляється сигмоїдною функцією, яка приймає будь-яке дійсне значення між 0 і 1.

Метод k-найближчих сусідів є одним з основних та важливих алгоритмів класифікації машинного навчання.

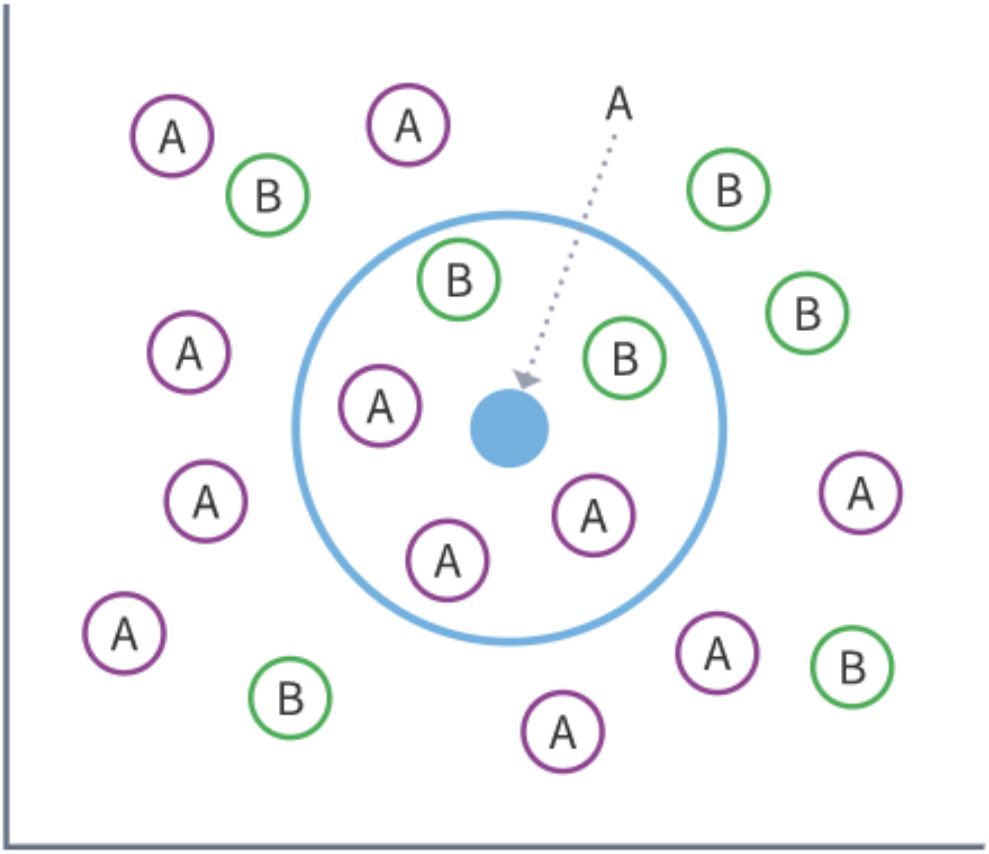


Рисунок 1.2 - Алгоритм k-найближчих сусідів

Цей метод є простим у реалізації алгоритмом машинного навчання під наглядом, який можна використовувати для вирішення як проблем класифікації, так і регресії. Алгоритм під наглядом машинного навчання, на відміну від непідконтрольного алгоритму машинного навчання, покладається на розмічені вхідні дані для вивчення функції, яка обробляє відповідні вихідні дані при отриманні нових нерозмічених даних [10]. Алгоритми машинного навчання, що контролюються, використовуються для вирішення проблем класифікації або регресії. Проблема класифікації має дискретне значення на виході. Алгоритм k-найближчих сусідів передбачає, що подібні речі існують у безпосередній близькості. Іншими словами, подібні речі близькі один одному. Алгоритм k-найближчих сусідів залежить від цього припущення, що є досить правдивим, щоб алгоритм був корисним. Цей метод фіксує ідею подібності, яку іноді називають дистанцією чи близькістю, з деякою математикою, з допомогою якої обчислюється відстань між точками на графіку. Алгоритм заключає у собі наступні кроки:

– завантаження даних;

– ініціалізація k до обраної кількості сусідів;

– для кожного прикладу обчислюється відстань між прикладом запиту та поточним прикладом даних;

– для них додається відстань та індекс прикладу до упорядкованої колекції;

– сортування упорядкованої колекції відстаней та показників від найменшої до найбільшої у порядку зростання за відстанями;

– далі обираються перші записи k із відсортованої колекції та отримуються мітки вибраних k записів;

– якщо розглядається задача регресії, повертається середнє значення міток k;

– якщо задача класифікації, повернеться режим k міток.

Щоб вибрати потрібне значення k, що відповідає обраним даним, алгоритм запускається кілька разів із різними значеннями k і вибирається k, що зменшує кількість помилок, зберігаючи здатність алгоритму точно робити прогнози, коли йому надані дані, які він не бачив раніше. Треба пам’ятати, що якщо значення K зменшувати до 1, як підсумок прогнози стають менш стабільними. І навпаки, у міру збільшення значення K прогнози стають більш стабільними завдяки усередненню, і, таким чином, більш ймовірно, що буде отримано більш точні прогнози до певного моменту. Врешті-решт можна буде спостерігати все більшу кількість помилок. Саме в цей момент буде зрозуміло, що значення K надто далеко від потрібного. У випадках, коли приймається більшість, наприклад, вибір режиму в проблемі класифікації, серед міток, K зазвичай робиться непарним номером, щоб мати перерву. Перевагами цього алгоритму є те, що він простий і легкий у здійсненні. Також не потрібно створювати модель, налаштовувати кілька параметрів або робити додаткові припущення. Алгоритм є універсальним. Його можна використовувати для класифікації, регресії та пошуку. Недоліком даного алгоритму є те, що він стає значно повільнішим у міру збільшення кількості прикладів та незалежних змінних. Основний недолік даного методу, коли він стає значно повільнішим, оскільки об'єм даних збільшується, робить його непрактичним вибором у середовищах, де прогнози потрібно робити швидко. Крім того, існують більш швидкі алгоритми, які дозволяють отримати більш точні результати класифікації та регресії. Однак, якщо у вас мається достатня кількість обчислювальних ресурсів для швидкого оперування з даними, які використовуються для прогнозування, алгоритм k-найближчих сусідів все ще може бути корисним для вирішення проблем, які мають рішення, які залежать від ідентифікації подібних об'єктів. Прикладом цього є використання алгоритму в системах рекомендацій, застосуванні K-пошуку.

Дерево рішень це ще один метод, який використовується в задачах класифікації.

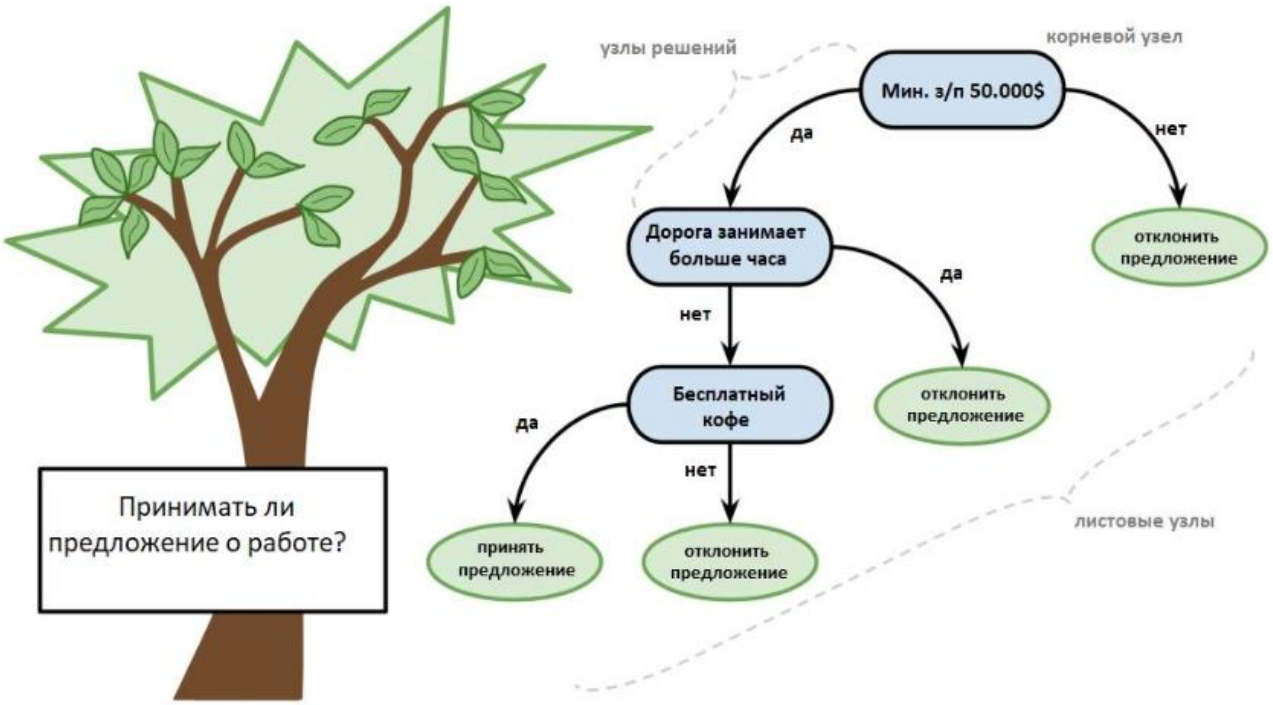


Рисунок 1.3 – Приклад дерева рішень

Це інструмент підтримки прийняття рішень, який використовує деревоподібний граф або модель рішень та їх можливі наслідки, включаючи результати випадкових подій, витрати на ресурси та корисність. Це один із способів відображення алгоритму, який містить лише умовні оператори управління. Дерево рішень представляє собою структуру, подібну до блок-схеми, у якій кожен внутрішній вузол являє собою тест. Як приклад можна навести випадок, коли монета перевертається на аверс чи реверс. Кожна гілка представляє результат тесту, а кожен вузол являє собою мітку класу, або рішення, прийняте після обчислення всіх атрибутів. Шляхи від кореня до листа представляють правила класифікації. Алгоритми навчання на основі дерев рішень вважаються одним з найкращих і в основному використовуються в методах навчання під наглядом. Методи дерева рішень розширюють можливості прогнозування моделей з високою точністю, стабільністю та простотою інтерпретації. На відміну від лінійних моделей, вони досить чітко відображають нелінійні зв’язки. Вони пристосовані для вирішення будь-яких

проблемних питань, таких як класифікація чи регресія. Одним із загальних термінів, які використовуються з деревами рішень, є корінний вузол. Він представляє всю сукупність чи зразок, і далі ділиться на два або більше однорідних наборів. Розщепленням називають процес поділу вузла на два або більше підвузлів. Вузол прийняття рішення представляє собою підвузол, який розбивається на інші підвузли. Листовий або кінцевий вузол це вузол, який не розщеплюються. Видалення підвузла вузла рішення, називають обрізкою. Протилежний цьому є процес розщеплення. Гілка або суб-дерево представлена підрозділом цілого дерева. Вузол, який розділений на підвузли, називається батьківським вузлом підвузлів, тоді як підвузли є дочірнім батьківським вузлом. Як результат, дерево прийняття рішень є одним із найпопулярніших алгоритмів класифікації, який використовується в майнінгу даних та машинному навчанні. Прикладом використання дерева рішень можна назвати допомогу з визначенням пріоритетності лікування пацієнтів невідкладної допомоги за допомогою прогнозної моделі, заснованої на таких факторах, як вік, артеріальний тиск, стать, місце розташування та тяжкість болю. Дерева рішень зазвичай використовуються в дослідженні операцій, зокрема в аналізі рішень, щоб допомогти визначити стратегію, найбільш ймовірну для досягнення мети. Завдяки своїй простоті, діаграми дерев використовуються в широкому діапазоні галузей та дисциплін, включаючи енергетику, фінансову галузь, інженерну, медичні послуги, фармацевтику, освіту, право та бізнес. Основними кроками розробки алгоритму дерева рішень є:

– розміщення найкращого атрибуту набору даних у корені дерева;
– поділ навчального набору на підмножини.

Ці два кроки повторюються для кожної підмножини, поки не буде знайдено вузли листя у всіх гілках дерева. У деревах рішень для прогнозування мітки класу для запису виконуються наступні дії:

– порівнюються значення кореневого атрибута з атрибутом запису;

– на основі порівняння вибирається гілка, що відповідає цьому значенню, і робиться перехід до наступного вузла;

– порівняння значення атрибутів вихідного запису з іншими внутрішніми вузлами дерева, поки не буде досягнено вузла із прогнозованим значенням класу.

Змодельоване дерево рішень можна використовувати для прогнозування цільового класу або значення. Існує деяка припущень, які робляться під час використання дерева рішень. На початку весь навчальний набір розглядається як корінь. Значення характеристик за краще вважають категоричними. Якщо значення безперервні, то перед побудовою моделі вони дискретизуються. Записи поширюються рекурсивно на основі значень атрибутів. Порядок розміщення атрибутів як кореневого чи внутрішнього вузла дерева виконується за допомогою певного статистичного підходу. Перевагами використання дерева рішень можна назвати легкість до розуміння. Вихід із дерева рішень дуже легко зрозуміти навіть людям з не аналітичним мисленням. Для їх зчитування та інтерпретації не потрібні будь-які статистичні знання. Його графічне зображення дуже інтуїтивно зрозуміле, і користувачі можуть легко співвідносити свою гіпотезу. Також цей метод є корисним при дослідженні даних. Дерево рішень є одним із найшвидших способів виявлення найбільш значущих змінних та зв’язку між двома чи більше змінними. За допомогою дерев рішень можна створити нові змінні та функції, які мають кращу потужність для передбачення цільової змінної. Його також можна використовувати на етапі дослідження даних. Наприклад, якщо йде праця над проблемою, коли є в наявності інформація, яка є у сотнях змінних, тут дерево рішень допоможе визначити найбільш значну змінну. Дерева рішень неявно виконують вибір функції. Також можна зазначити, що цей метод вимагає порівняно невеликих зусиль від користувачів для підготовки даних. Потрібно менше очищення даних, тобто цей алгоритм вимагає меншої кількості очищення даних порівняно з іншими методами моделювання. Тип даних не є

обмеженням. Так дерево рішень може обробляти як числові, так і категоричні змінні. Також цей алгоритм може вирішувати проблеми з кількома виходами. Дерево рішень вважається непараметричним методом. Це означає, що дерева рішень не мають припущень щодо розподілу простору та структури класифікатора. Нелінійні зв'язки між параметрами не впливають на продуктивність дерева. А кількість гіпер-параметрів, які слід настроїти, є несуттєвою і прямує до нуля. Серед недоліків метода дерева рішень можна назвати перенавчання. Надмірна обробка є однією з найбільш практичних труднощів для моделей дерева рішень. Ця проблема вирішується шляхом встановлення обмежень на параметри моделі та обрізки. Дерево рішень не підходить для безперервних змінних. Працюючи з безперервними числовими змінними, дерево рішень втрачає інформацію, коли воно класифікує змінні в різні категорії. Дерева рішень можуть бути нестабільними, оскільки невеликі зміни даних можуть призвести до генерування зовсім іншого дерева. Це називається дисперсією, яку потрібно зменшити такими методами, як бегінг та бустінг. Також є важливим, що розрахунки можуть стати складними, коли існує багато міток класу. На цьому етапі можна задатись питанням, що якщо можна використовувати логістичну регресію для проблем класифікації, а лінійну регресію для проблем регресії, тоді чому існує потреба у використанні дерев. Відповідаючи на це, можна сказати, що, власне, можна використовувати будь-який алгоритм. Це залежить від типу проблеми, яка вирішується. Деякими основними факторами, які допоможуть у прийнятті рішення, який алгоритм використовувати, є умова, що якщо залежність між залежною та незалежною змінною добре наближена лінійною моделлю, лінійна регресія випереджає дерево-модель. Але якщо існує велика не лінійність і складна залежність між залежними та незалежними змінними, модель дерева перевершить класичний метод регресії. Щоб побудувати модель, яка легка до розуміння, модель дерева рішень завжди буде краще зрозумілою, ніж лінійна модель. Моделі дерев рішень інтерпретувати навіть простіше, ніж результати лінійної регресії.

Підсумовуючи, треба зафіксувати, що не всі проблеми можна вирішити лінійними методами. Було помічено, що моделі на основі дерев змогли ефективно відобразити не лінійність. Такі методи, як дерева рішень широко використовуються у всіх видах проблем науки про дані. Алгоритм дерева рішень належить до сімейства керованих алгоритмів навчання. На відміну від інших керованих алгоритмів навчання, алгоритм дерева рішень може використовуватися і для вирішення проблем регресії та класифікації. Загальний мотив використання дерева рішень полягає у створенні навчальної моделі, яка може використовуватись для прогнозування класу чи значення цільових змінних, вивчаючи правила прийняття рішень, виведені з попередніх навчальних даних. Основна проблема в реалізації дерева рішень полягає у визначенні, які атрибути потрібно розглядати як кореневий вузол та кожен рівень. Дерева рішень часто імітують мислення на рівні людини, тому зрозуміти дані і зробити хороші тлумачення не спричиняє проблем. Ефективний розподіл на основі максимального отримання інформації є ключовим для класифікатора дерева рішень. Однак у реальному світі, де мільйони даних не розділяються на чисті класи, це практично неможливо або може зайняти довший час навчання, і тому треба зупинитись на точках у вузлах дерева.

Інший алгоритм, який використовується для задач бінарної класифікації є метод опорних векторів.

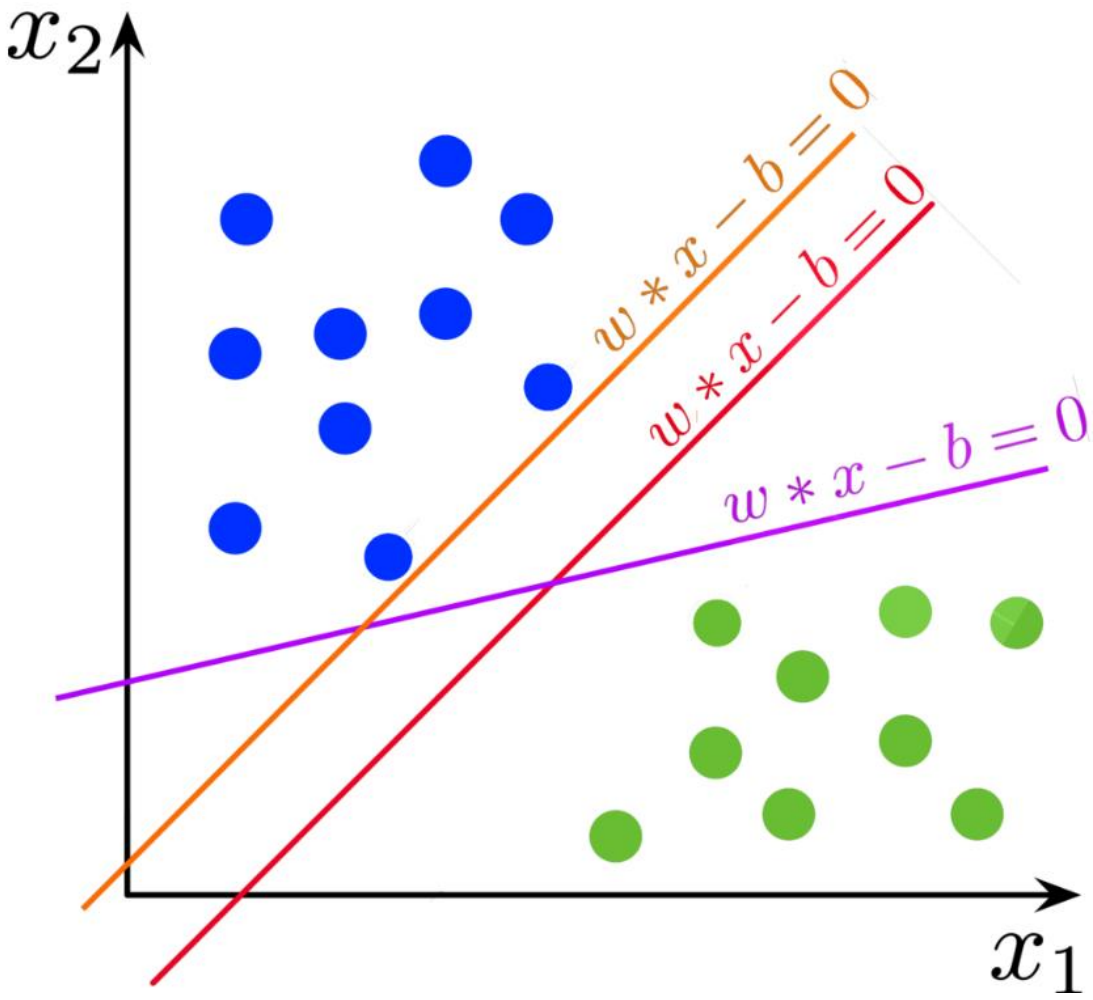


Рисунок 1.4 – Метод опорних векторів

Завданням алгоритму опорних векторів є знаходження гіперплану у N-мірному просторі, де N - кількість ознак, що чітко класифікує точки даних. Для розділення двох класів точок даних існує багато можливих гіперпланів, які можна вибрати. Метою є знаходження площини, яка має максимальний запас, тобто максимальну відстань між точками даних обох класів. Максимальне збільшення відстані забезпечує певне підкріплення, щоб майбутні точки даних можна було класифікувати з більшою впевненістю. Гіперплани - це межі рішення, які допомагають класифікувати точки даних. Точки даних, що падають з будь-якої сторони гіперплана, можна віднести до різних класів. Також розмірність гіперплана залежить від кількості ознак. Якщо кількість вхідних функцій дорівнює двом, то гіперплан - це лише лінія. Якщо кількість вхідних ознак дорівнює трьом, то гіперплан стає двовимірною площиною.

Важко уявити, коли кількість функцій перевищує три. Векторами опори називають точки даних, які знаходяться ближче до гіперплану і впливають на положення та орієнтацію гіперплана. Використовуючи ці вектори підтримки, максимально збільшується запас класифікатора. Видалення опорних векторів змінить положення гіперплана. У методі опорних векторів розглядається вихід лінійної функції, і якщо цей вихід більший за 1, це визначається одним класом, і якщо вихід -1, це визначається з іншим класом. Так як порогові значення змінюються на 1 і -1 в методі опорних векторів буде отримано діапазон значень від мінус одного до одиниці, який виступає в якості границі. Вартість дорівнює 0, якщо прогнозоване значення та фактичне значення мають однаковий знак. Якщо їх немає, то обчислюється значення втрат. Також додається параметр регуляризації функції витрат. Завданням параметра регуляризації є збалансування максимізації маржі та втрат. Тоді, коли функція втрат, можна брати часткові похідні ваг, щоб знайти градієнти. За допомогою градієнтів можна оновлювати ваги. Коли немає помилкової класифікації, тобто модель правильно передбачає клас точки даних, залишається лише оновити градієнт від параметра регуляризації. Коли є помилкова класифікація, тобто модель помиляється при прогнозуванні класу точки даних, тоді включаються втрати разом з параметром регуляризації для оновлення градієнта. Метод опорних векторів є елегантним і потужним алгоритмом.

Наївний Байєс є ще одним алгоритмом, який використовується для задач бінарної класифікації. Це методика класифікації, заснована на теоремі Байєса з припущенням незалежності серед прогнозів. Простіше кажучи, класифікатор припускає, що наявність певної особливості в класі не пов'язана з наявністю будь-якої іншої особливості.

Рисунок 1.5 – Алгоритм Наївного Байєсу

Модель проста у створенні та особливо корисна для дуже великих наборів даних. Поряд із простотою, цей метод, як відомо, перевершує навіть дуже складні методи класифікації. Перевагами цього алгоритму є те, що він легко і швидко передбачає клас набору тестових даних. Він також чудово працює в багато класовому передбаченні. Коли припускається незалежність ознак, класифікатор наївний Байєс працює краще порівняно з іншими моделями, такими як логістична регресія, і потрібно менше даних для навчання. Він добре працює у випадку категоричних вхідних змінних порівняно з числовими змінними. Для числових змінних передбачається нормальний розподіл. Недоліками цього підходу є те, що якщо категоріальна змінна має категорію у тестовому наборі даних, яка не спостерігалась у наборі даних тренувань, то модель призначить імовірність нуль і не зможе зробити прогнозування. Це часто називають "нульовою частотою". Для вирішення цього питання можна використовувати техніку розгладження. Один з найпростіших методів розгладження називається оцінкою Лапласа. З іншого боку, наївний Байєс також відомий як поганий оцінювач, тому ймовірні результати з

прогнозування не слід сприймати занадто серйозно. Ще одне обмеження це припущення незалежних прогнозів. У реальному житті майже неможливо отримати набір прогнозів, які є абсолютно незалежними. Наївний Байєс використовується у багатьох типах задач. Серед них є прогноз у реальному часі. Цей алгоритм також добре відомий для функції багатокласового передбачення. Також його використовують для класифікації тексту, фільтрування спаму, при аналізі почуттів, як наприклад в аналізі соціальних медіа для виявлення позитивних та негативних настроїв клієнтів. Також цей метод використовують у системах рекомендацій: класифікатор Байєса і фільтрація спільно будують систему рекомендацій, яка використовує методи машинного навчання для фільтрації інформації та прогнозування, чи сподобається користувачеві певний ресурс чи ні.

При розробці програмної системи використовувалась нейронна мережа. Нейронна мережа є одним із декількох алгоритмів машинного навчання, які використовуються при вирішенні класифікаційних проблем. Перевагою використання є здатність динамічно створювати складні функції прогнозування та імітувати людське мислення таким чином, який не може зробити жоден інший алгоритм.

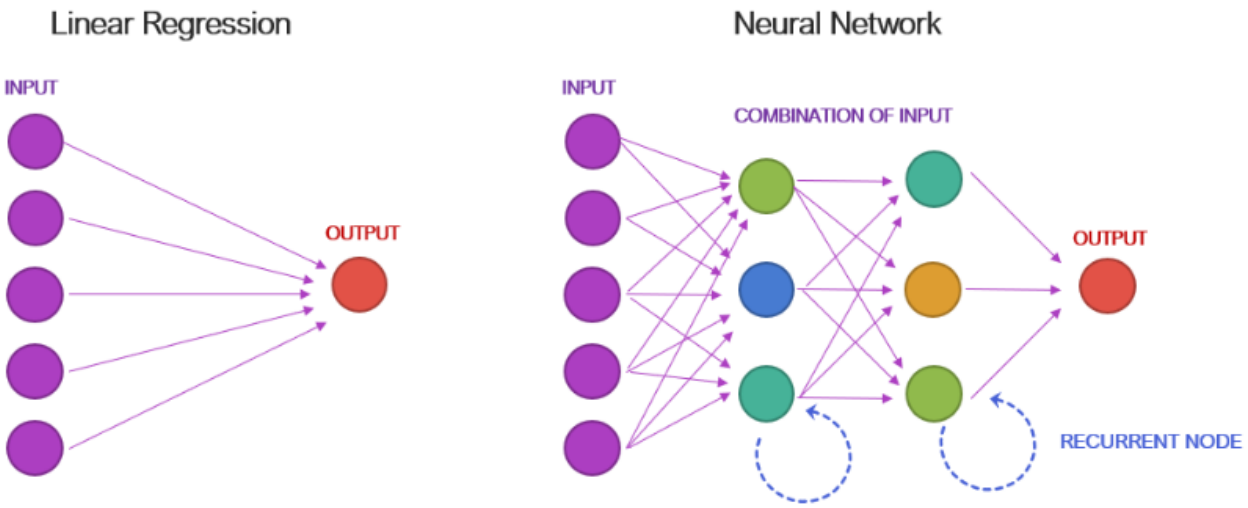


Рисунок 1.6 – Порівняння нейронної мережі та лінійної регресії

Існує багато проблем класифікації, для яких нейронні мережі дають найкращі результати. Штучні нейронні мережі побудовані з простих елементів, званих нейронами, які приймають реальну цінність, множать її на вагу і запускають її через нелінійну функцію активації. Побудувавши кілька шарів нейронів, кожен з яких отримує частину вхідних змінних, а потім передає свої результати наступним шарам, мережа може засвоїти дуже складні функції. Цей підхід є дуже ефективним для задач високої розмірності, здатний вирішувати складні взаємозв’язки між змінними, наборами категорій і складними функціями, що стосуються введення змінних на вихід. Інструментарій дозволяє настроювати параметри для запобігання надмірній та недостатній підгонці. Робота нейронної мережі не інтуїтивно зрозумілою і вимагає налаштування у процесі моделювання. У деяких випадках для ефективності необхідний великий навчальний набір. Процес створення передбачає відстеження прогресу в декількох експериментах та зберігання метрик і параметрів. Експерименти машинного навчання, особливо нейронні мережі, потребують постійних спроб та помилок, щоб правильно виправити модель. Модель нейронної мережі навчається, розуміючи, що допущена помилка, повертаються на кожній епосі в алгоритм, щоб виявити, які значення та з'єднання зробили вихід невірним.

1.4 Аналіз вимог до програмного забезпечення

Створена програмна система має забезпечувати наступні функції для користувача:

– можливість завантаження текстового файлу;

– отримання результатів щодо статі автора завантаженого тексту;

– можливість перегляду кольорового спектру завантаженого тексту;

– можливість перегляду частотних характеристик використаних назв кольорів.

Програмна система, що розроблюється, має надавати наступні варіанти використання:

Таблиця 1.1 – Варіант використання UC-1

Унікальний ідентифікатор	UC-1
Назва	Завантаження текстового файлу
Опис	Користувач завантажує текстовий файл у форматі txt
Учасники	Користувач
Передумови	Відкрита веб-сторінка для завантаження, наявність текстового файлу
Постумови	Користувач успішно завантажив файл
Сценарій	– Користувач натискає кнопку для завантаження файлу – Користувач додає файл – Натиснення кнопки «Завантажити»

Таблиця 1.2 – Варіант використання UC-2

Унікальний ідентифікатор	UC-2
Назва	Отримання результатів щодо статі автора
Опис	Після обробки тексту на екрані відображається отриманий результат
Учасники	Користувач
Передумови	Успішне завантаження файлу
Постумови	Відображений результат визначення статі автора
Сценарій	Користувач натискає кнопку «Старт»

Таблиця 1.3 – Варіант використання UC-3

Унікальний ідентифікатор	UC-3
Назва	Перегляд кольорового спектру
Опис	Після обробки тексту на екрані відображається палетка спектру використаних кольорів в тексті
Учасники	Користувач
Передумови	Успішне завантаження текстового файлу
Постумови	Відображена палетка спектру використаних кольорів
Сценарій	1. Користувач натискає кнопку «Старт»

Таблиця 1.4 – Варіант використання UC-4

Унікальний ідентифікатор	UC-4
Назва	Перегляд частотних характеристик використаних назв кольорів
Опис	На екрані відображаються графіки використання кольорів у тексті
Учасники	Користувач
Передумови	Успішне завантаження текстового файлу
Постумови	Відображені графіки
Сценарій	Користувач натискає кнопку «Старт»

Об’єднаємо сценарії використання у структурній схемі варіантів використання, яка зображена на Рисунку 1.7.

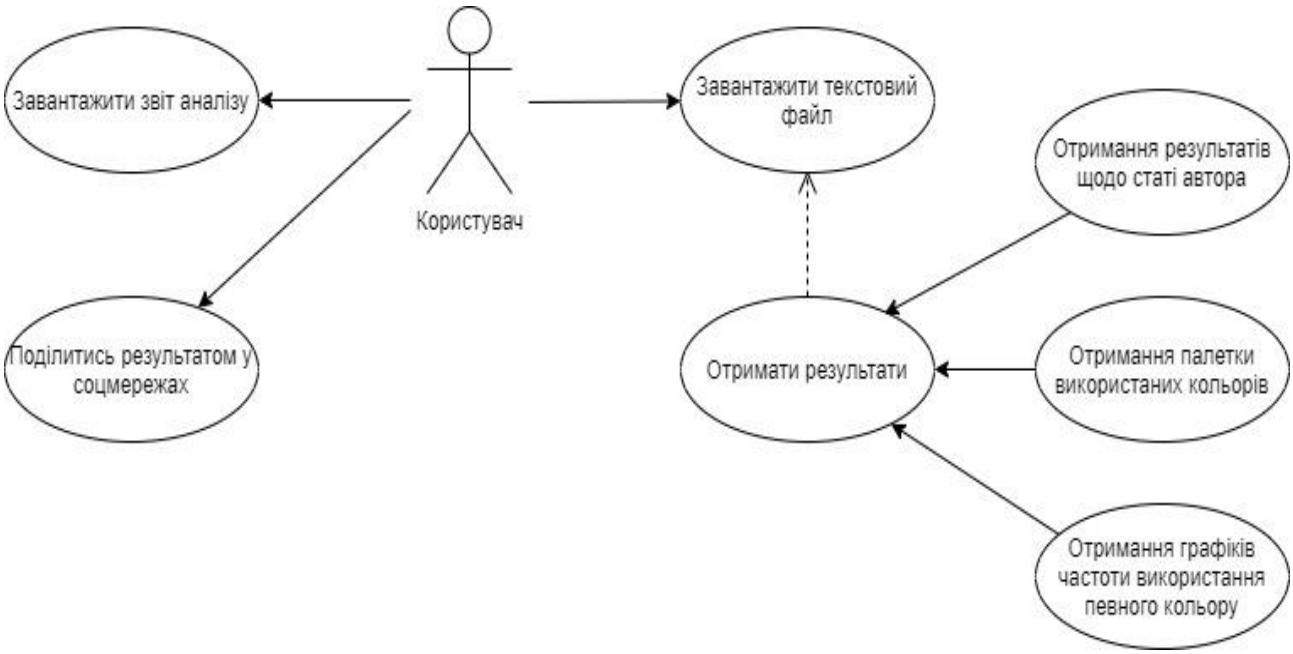


Рисунок 1.7 – Схема структурна варіантів використання

1.4.1 Розроблення функціональних вимог

Функціональні вимоги наведені у наступних таблицях:

Таблиця 1.5 – Варіант використання RQ-1

Унікальний ідентифікатор	RQ-1
Назва	Завантаження текстового файлу
Опис	Система має надавати можливість завантаження текстового файлу

Таблиця 1.6 – Варіант використання RQ-2

Унікальний ідентифікатор	RQ-2
Назва	Визначення статі автора
Опис	Система має визначити стать автора з певною вірогідністю

Таблиця 1.7 – Варіант використання RQ-3

Унікальний ідентифікатор	RQ-3
Назва	Відображення кольорового спектру
Опис	Система має відобразити палетку використаних кольорів у завантаженому тексті

Таблиця 1.8 – Варіант використання RQ-4

Унікальний ідентифікатор	RQ-4
Назва	Відображення частотних характеристик використаних назв кольорів
Опис	Система має відобразити графіки, які ілюструють частоту використання кольорів

Залежності між варіантами використання та функціональними вимогами відображено на матриці трасування, що зображена на Рисунку 1.8.

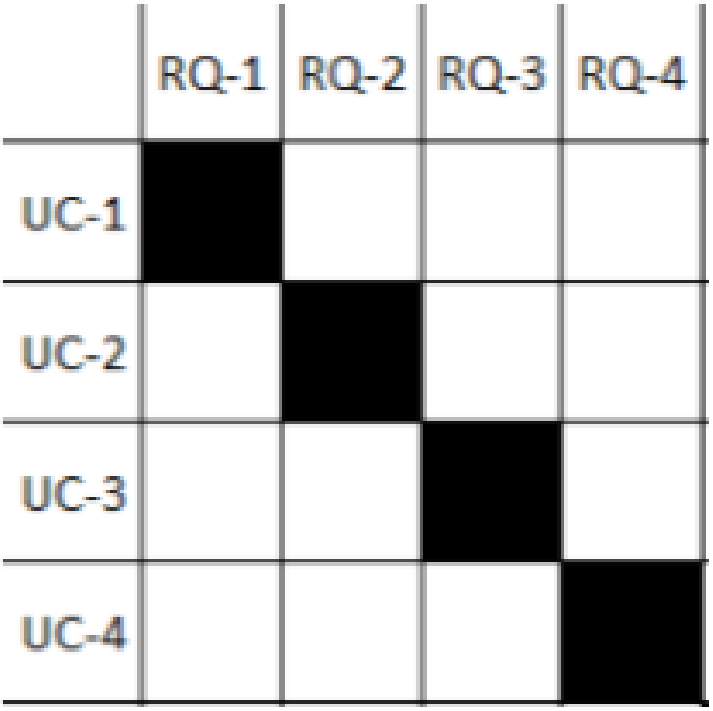


Рисунок 1.8 – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Спроектована програмна система для визначення статі автора художнього твору представлена веб-застосунком. Дана система побудована на основі машинного навчання. Клієнтська частина потребує наявності одного з перечислених Web - браузерів: Mozilla Firefox, Google Chrome, Internet Explorer, Safari, Opera Browser. Мова додатку: українська. Система повинна мати інтуїтивно зрозумілий інтерфейс користувача, елементи додатку повинні бути однозначними і зрозумілими. Отримані результати повинні забезпечувати точність близько 80 відсотків.

1.4.3 Постановка комплексу завдань модулю

Метою розробки даної програмної системи є можливість визначення статі автора у художніх творах. Це виконується шляхом аналізу використання назв кольорів тим чи іншим, невідомим для системи, автором. Даними для навчання

моделі є художні твори. Під час аналізу розроблених вимог були сформульовані такі задачі розробки:

– створення словника, який буде містити відповідні кольори або аналогії;

– складання датасета для навчання;

– створення системи графіків, яка зможе візуально представити результат досліджень;

– можливість завантаження текстового файлу;

– інтелектуальний аналіз текстів.

1.5 Висновки по розділу

У даному розділі було проведено детальний аналіз та опис предметної області, були розроблені варіанти використання та побудовані функціональні та нефункціональні вимоги до розроблюваної програмної системи. Також були розроблені таблиці варіантів використання, функціональні вимоги та матриця залежностей варіантів використання та функціональних вимог. Вони допоможуть покращити процес розробки. Були розглянуті та досліджені вже існуючі технічні рішення для вибору ефективних методів та алгоритмів.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для визначення процесів, які необхідно реалізувати у веб-застосунку, було використано нотацію опису бізнес-процесів користувачів системи – BPMN. Були змодельовані основні бізнес-процеси, що відбуваються у системі. Ці процеси включають:

- завантаження текстового файлу у систему;
- обробка завантаженого файлу;
- пошук у словнику;
- обробка даних;
- формування результатів;
- отримання результатів.

Структурна схема бізнес-процесів наведена на Рисунку 2.1.

Користувач завантажує текстовий файл у систему. Отриманий файл проходить попередню обробку. Якщо кількість слів є меншою за необхідну про це повідомляється користувачеві. Далі отриманий файл проходить попередню обробку. Наступним кроком є пошук за словником використаних кольорів та обробка отриманих даних. На цій підставі формуються результати та демонструються користувачеві.

Рисунок 2.1 – Схема бізнес-процесу

Основними завданнями цієї роботи є:

– створення словнику кольорів;

– складання датасету для навчання;

– коректне та успішне навчання моделі, та виходячи з цього, коректна категоризація даних;

– відображення використаних кольорів у вигляді палетки кольорів;

– відображення частотного застосування назв кольорів.

Для роботи з моделлю та побудови нейронної мережі використана бібліотека для машинного навчання Keras. При створенні словника та складанні датасету додатково були розроблені два парсери з використання бібліотеки Requests та lxml.

2.1.1 Робота з кольорами

Використання назв кольорів авторами є визначальною характеристикою при створенні моделі. Для відображення кожного з кольорів використана кольорова модель, яка визначає спосіб кодування кольору за допомоги трьох кольорів, а саме RGB модель [11]. Вибір цієї моделі зумовлений тим, що RGB більш придатною кольоровою моделлю для відображення на екрані, у той час коли CMYK модель є більш придатною для печаті. Значення кольорів RGB підтримуються у всіх браузерах. Кожен параметр цієї моделі визначає інтенсивність кольору як ціле число від 0 до 255. Створений словник кольорових означень складається з назв кольорів, їх аналогій та їх RGB кодування. Він складається близько з 200 найменувань. У веб застосунку знайдені кольори відображаються як показано на Рисунку 2.2.

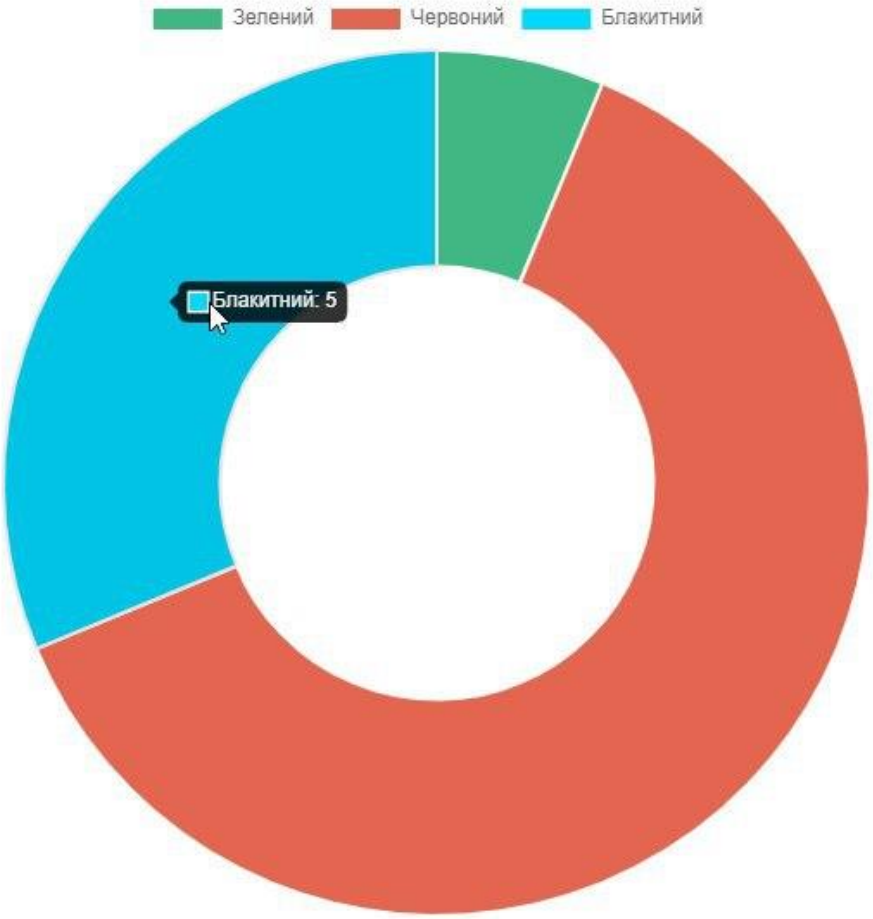


Рисунок 2.2 – Відображення кольорів

У сформованому звіті з результати аналізу кольори зібрані у таблицю, яка включає колір, його назву, кодування RGB.

2.1.2 Парсинг даних сайтів з творами

На початку веб сторінка зчитується за допомогою бібліотеки Requests.

```
import requests
	user_id = 12345
	url = 'http://www.supertext.lib/user/%d/votes/list/ord/date/page/2/#list' % (user_id) # url для другої сторінки
	r = requests.get(url)
	with open('test.html', 'w') as output_file:
	  output_file.write(r.text.encode('cp1251'))
```

Далі текст цієї веб-сторінки розбирається за допомогою бібліотеки lxml та мови Xpath. Наприклад так:

```
from lxml import html
test = '''
   <html>
      <body>
         <div class="first_level">
            <h2 align='center'>one</h2>
            <h2 align='left'>two</h2>
         </div>
         <h2>another tag</h2>
      </body>
   </html>
'''
tree = html.fromstring(test)
tree.xpath('//h2') # все h2 теги
tree.xpath('//h2[@align]') # h2 теги с атрибутом align.
tree.xpath('//h2[@align="center"]') # h2 теги с атрибутом align рівним "center"
div_node = tree.xpath('//div')[0] # div тег
div_node.xpath('.//h2') # все h2 теги, які є дочірними div ноді
```

Для того щоб створити датасет, треба ітерувати всі можливі сторінки із творами на всіх доцільних сайтах та зберігати у базу даних з іменем автора. Додатково треба парсити вікіпедію, щоб зазначити стать автора. Додатково спаршено список кольорів з сайту Вікіпедії [12].

2.2 Архітектура програмного забезпечення

Для розробки програмної системи була обрана мова програмування Python завдяки наступним властивостям:

– зрозумілість, Python насправді простий для розуміння, тому його легко використовувати;

– структурованість, зосереджений на читанні коду та є універсальною та добре структурованою мовою;

– універсальність, Python можна використовувати для вирішення майже будь-яких задач, це надзвичайно корисно для розробки веб-застосунків, штучного інтелекту та аналізу даних;

– простота інтеграції, у багатьох місцях Python використовується як мова інтеграції наявних компонент, його легко інтегрувати з іншими мовами програмування;

– масштабованість, ця мова програмування є пристосованою для зручного масштабування;

– легкість створення прототипів, має можливість швидко розробляти веб-застосунки, вимагає менше кодування, розробка прототипів економить час;

– велика колекція вбудованих бібліотек, пропонує велику кількість вбудованих бібліотек, які можна зручно використовувати для маніпулювання даними, обміну даними та машинного навчання;

– підтримка машинного навчання, велика екосистема бібліотек є однією з основних причин, чому Python є кращим для машинного навчання, воно вимагає постійної обробки даних, і щоб зробити це ефективно, бібліотеки Python надають доступ для обробки та трансформації даних;

– пропонує гнучкість, що надалі дозволяє вибирати стилі програмування, які зручніші для використання для вирішення різних типів проблем найбільш продуктивно.

Програмна система створена на основі нейронної мережі. Схема нейронної мережі наведена на Рисунку 2.3.

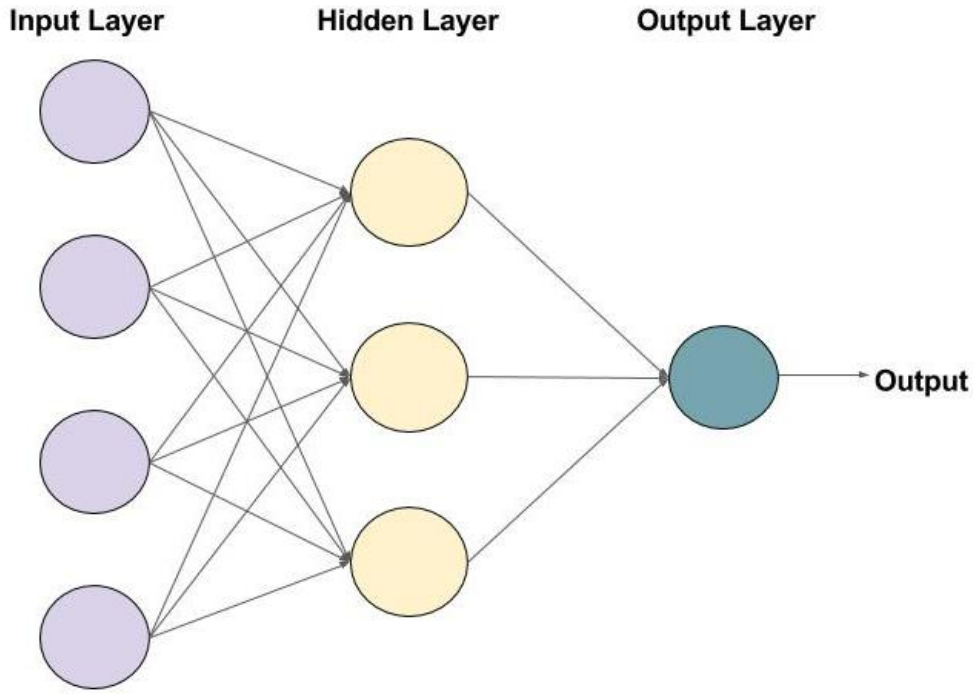


Рисунок 2.3 – Нейронна мережа

Нейронна мережа складається з вузлів, які з'єднані як показано вище. Датасет для навчання містить мітки статі авторів та обрані тексти художніх творів.

Keras - API високого рівня, який використовується для побудови та навчання моделей глибокого навчання [13]. Keras здатний працювати поверх TensorFlow. Він розроблено з акцентом на швидке експериментування. Модель розуміється як послідовність або граф автономних, повністю настроюваних модулів, які можуть бути підключені разом з мінімальними обмеженнями. Keras є зручним для швидкого складання прототипів, сучасних досліджень та виробництва з наступними перевагами:

– зручний для користувачів, має простий та стійкий інтерфейс, оптимізований для випадків загального використання, забезпечує чіткі та корисні відгуки про помилки користувачів;

– модульність, моделі Keras створюються шляхом з'єднання модулів, які можна підлаштовувати під конкретну задачу, разом, з невеликими обмеженнями;

– легко розширюється, надає можливість створення нових шарів, метрик та функції втрат;

– мінімалізм, кожен модуль повинен бути коротким і простим;

– масштабованість, нові модулі можуть бути легко додані і розширені;

– багатошаровість, Keras має ряд попередньо вбудованих шарів;

– працює безперебійно на CPU та GPU.

TensorFlow Framework - це популярна система машинного навчання з відкритим кодом для підготовки нейронної мережі для програм, які використовують машинне навчання [14]. Спочатку розроблена командою Google Brain Team для проведення машинного навчання та глибоких досліджень нейронних мереж. Достатньо загальна, щоб застосувати і в широкому спектрі інших областей. TensorFlow пропонує широкий набір функцій та класів, які дозволяють користувачам будувати різні моделі з нуля. TensorFlow відокремлює визначення обчислень від їх виконання. Базовий клас оптимізатора забезпечує методи обчислення градієнтів для втрат та застосування градієнтів до змінних. Колекція підкласів реалізує класичні алгоритми оптимізації, такі як GradientDescent та Adagrad. TensorFlow надає функції для обчислення похідних для обчислювального графа TensorFlow, додаючи операції з графом. Цей фреймворк має наступні особливості:

– простота побудови моделі, дозволяє легко створювати та навчати ML-моделі, використовуючи інтуїтивні API високого рівня, такі як Keras, з швидким виконанням, що дозволяє робити негайну ітерацію моделі та просте налагодження;

– легке навчання та розгортка моделі в хмарі на попередньому етапі, у браузері чи на пристрої незалежно від використаної мови;

– проста і гнучка архітектура для переходу нових ідей від концепції до коду та швидшої публікації.

В якості бази даних було обрано MongoDB. Для веб застосунку буде використано Dash, який використовує Flask та React.js. Dash має багато шаблонів графіків для статистичних даних. Планується виведення деяких статистичних даних, тому використання Dash доцільне.

Додаток міститиме поле для завантаження текстового документу на сервер. Присутня кнопка «Старт», яка розпочинає аналіз. Після аналізу користувачеві будуть показані діаграми використаних кольорів у творі, інші графіки. Також виводиться відповідь про ймовірну стать автора та точність цієї відповіді.

2.3 Конструювання програмного забезпечення

Як зазначалось вище, обчислювальні процеси зосереджені у нейронній мережі. Вона має наступну структуру:

– початок (вхідні вузли);
– прихований шар;
– шар вихідних даних.

Значення з вхідного вузла подаються вперед до прихованого шару. При кожному з’єднанні виконується функція активації і в кінці вихідний шар з одним вихідними вузлом. У даному випадку задачі бінарної класифікації це один вузол. Задля вирішення даної задачі використовується послідовна модель. Послідовна модель – це лінійний стек шарів. Функція активації для прихованого шару – rectifier, а для вихідного – сігмоїда. Під час будування послідовної моделі виконуються наступні кроки:

– додавання шару за шаром;
– налаштовування процесу навчання;
– визначення оптимізатору;
– визначення функції втрат.

Перш ніж розпочати з навчання моделі, налаштовується процес навчання. Це робиться методом compile. Цей метод визначає оптимізатор і функцію втрат.

Використовуються методи оптимізації, щоб зменшити похибку між обчислюваним та бажаним результатами. Помилка визначається функцією втрат (loss function), втрати якої мінімізуються за допомогою оптимізатора. Оскільки тренування є ітераційним процесом, тренування не буде припинятися після його проведення. Тому вказується кількість ітерацій, для яких буде навчатися модель. Закінчені ітерації прийнято називати епохами. Визначення потрібної кількості ітерацій є важливим етапом. Вони запускаються деяку кількість разів, щоб можна було побачити, як змінюються втрати та точність тренувань після кожної епохи. Далі використовується метод evaluate для вимірювання точності моделі. Це робиться як для даних навчання, так і для даних тестування.

Після етапу витягу інформації з датасету, яка містить тексти художніх творів та позначки статі авторів, використовується BOW (bag of words) model, щоб перетворити тексти творів на вектори. Використовується метод CountVectorizer. Твори розділяються навпіл на навчальну вибірку, відповідно для навчання моделі та тестову вибірку для перевірки моделі. Ці вибірки є збалансованими, тобто містять однакову кількість авторів чоловіків та жінок. Ці два набори дозволяють оцінити точність і побачити, чи здатна модель добре працювати на даних, яких раніше не бачила. Також це спосіб переконатися, що модель не перенавчена, оскільки це означатиме, що модель здебільшого просто запам'ятовувала дані тренувань. Це пояснювало б велику точність даних про навчання, але низьку точність даних тестування.

На Рисунку 2.4 зображено послідовність маніпулювань з моделлю під час навчання.

Рисунок 2.4 – Послідовність дій мережі

Словник кольорів містить назви кольорів та їх аналогії. Він представлений двома таблицями бази даних.

Таблиця 2.1 – Опис сутності Colors

Стовбець	Тип	Первинний ключ	Зовнішній ключ	Опис
id	int	+	-	Унікальний ідентифікатор
name	string	-	-	Назва кольору
rgb	int	-	-	Кодування RGB

Таблиця 2.2 – Опис сутності Color_analogies

Стовбець	Тип	Первинний ключ	Зовнішній ключ	Опис
id	int	+	-	Унікальний ідентифікатор
parent_id	int	-	+	Зовнішній ключ для визначення аналогій
name	string	-	-	Назви аналогій кольорів

Для тренування моделі було відібрано твори авторів жінок та чоловіків.

Таблиця 2.3 – Опис сутності Texts

Стовбець	Тип	Первинний ключ	Зовнішній ключ	Опис
id	int	+	-	Унікальний ідентифікатор
gender	boolean	-	-	Позначка статі
text	char	-	-	Текст художнього твору

Додано специфікацію методів.

Таблиця 2.4 – Специфікація методів

Метод	Вхідні дані	Вихідні дані	Опис
checkFileExtention	Файл		Перевіряє коректність розширення файлу
sendForGenderAnalisys	Файл		Відправка файлу для аналізу
getAnalisysList		Список минулих результатів	Отримує список результатів
shareToFacebook	Результати аналізу		Додає ссилку з результатами у ленту соцмережі
downloadReport		Файл pdf з результатами	Ініціює завантаження файлу звіту
fc_model		Оновлений стек шарів	Додає екземпляр шару поверх стеку шарів
s_model		Модель	Створює екземпляр роздільної моделі
train_sequence_model	Кількість зразків на партію, масив для зважування функції втрат.		Запуск тренування моделі послідовності для даного набору даних

Продовження таблиці 2.4

Метод	Вхідні дані	Вихідні дані	Опис
get_layer	Назва шару, індекс	Екземпляр шару	Отримує шар на основі його індексу
_evaluate	Кількість зразків на партію, масив для зважування функції втрат, загальна кількість кроків	Скалярні тестові втрати	Повертає значення втрати та показники для моделі
_predict	Кількість зразків на партію, загальна кількість кроків, максимальний розмір для черги генератора, максимальна кількість процесів	Масив прогнозів	Згенерує прогнози виводу для вхідних зразків
load_dataset	Шлях до каталогу даних, насіння	Набір даних про навчання та валідацію	Завантажує набір даних для навчання та валідації

Продовження таблиці 2.4

Метод	Вхідні дані	Вихідні дані	Опис
_make_dir	path		Створення каталогу
fit	Розмір партії, кількість епох, дані для тренування, дані для перевірки	Запис значень втрат у навчанні та показників метрики в наступні епохи, а також значення втрат валідації та значення метрики валідації	Тренує модель для фіксованої кількості епох (ітерацій на наборі даних)
train_on_batch	TensorFlow tensor, цільові дані, ваги	Скалярні втрати при навчанні	Виконує одне оновлення градієнта на одній партії даних
test_on_batch		Скалярні тестові втрати	Тестує модель на одній партії зразків
run_eagerly		Значення true or false	Визначає, чи модель повинна прагнути запуску

2.4 Аналіз безпеки даних

Клієнт-серверна взаємодія відбуватиметься через протокол HTTPS, що є розширенням протоколу HTTP з підтримкою криптографічного шифрування. Він використовує криптографічні протоколи SSL та TLS. HTTPS не є самостійним протоколом, це просто HTTP, що використовує криптографічні протоколи. HTTPS достатньо, щоб захистити дані від найрозповсюдженіших атак. За замовчуванням не дозволено додатку надсилати запити через HTTP, і примусово вимагає HTTPS.

Додаток не має ніякої аутентифікації чи конфіденційних даних, тому єдине що треба захистити – це сервер. DDOS атаку, яка може зайняти усі обчислювальні ресурси системи, можна попередити виставивши максимальне число запитів з хоста за період часу.

2.5 Висновки по розділу

В даному розділі було описано основні технології, засоби, що використовуються в розроблюваній системі. Була наведена та описана схема бази даних. Було також проаналізовано безпеку системи. Наведений змістовний опис процесу навчання моделі та використані методи.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Аналіз і забезпечення якості розглядається на всіх етапах життєвого циклу програмного продукту. Процес досягнення якості програмного забезпечення залежить від процесу проектування, який повинен бути покроковим та широким і включати систематизований набір дій із забезпечення відповідності вимогам продуктом, який є створеним відповідно до поставлених технічних задач. При цьому верифікація та валідація програмного забезпечення належать до управління якістю і є важливими процесами забезпечення якості програмного продукту на всіх етапах його життєвого циклу.

Тестування програмного забезпечення є невід'ємною частиною розробки програмного забезпечення. Метою тестування програмного забезпечення є підвищення ймовірності того, що додаток, призначений для тестування, буде працювати правильно при будь-яких обставинах, підвищити ймовірності того, що додаток, призначене для тестування, буде відповідати всім описаним вимогам та надання актуальної інформації про стан продукту на даний момент.

3.1 Опис процесів тестування

Об'єктом тестування є:

– нейронна мережа;

– веб застосунок.

Компоненти, що будуть тестуватись:

– визначення статі нейронною мережею;

– завантаження файлу через веб-інтерфейс;

– запуск обчислення;

– отримання головного результату, а саме статі;

– отримання додаткових даних для графіків.

Тестування відбуватиметься у режимі білої скриньки, оскільки тестування буде проведено розробником. Також буде проведено негативне тестування.

Будуть проведені наступні типи тестів:

– тестування інтерфейсу користувача;
– тестування роботи обчислювальної системи;
– тестування зручності використання.

Тести виконуються розробником у ручному режимі. Автоматичне тестування веб-застосунку недоцільне, тому що його функціонал дуже простий.

3.2 Опис контрольного прикладу

У наступних таблицях будуть описані проведені тести:

– тестування інтерфейсу користувача;
– завантаження тексту через веб-застосунок;
– запуск обчислення;
– отримання даних від back-end(у) застосунку;
– відсутність ймовірності запуску обчислень без завантаження файлу;
– при завантаженні файлу, який містить недостатню кількість слів для обчислень, повідомити про це користувача;
– завантаження звіту аналізу.

Таблиця 3.1 – Завантаження тексту через веб-застосунок

Мета тесту	Завантаження текстового файлу
Передумови	Файл у форматі txt
Схема виконання	Надати файл до поля
Очікуваний результат	Файл завантажується на сервер
Дійсний результат	Файл завантажується

Таблиця 3.2 – Запуск обчислення

Мета тесту	Запуск обчислення
Передумови	Файл збережений на сервері
Схема виконання	Натиснути кнопку на сайті
Очікуваний результат	Початок процесу обчислення
Дійсний результат	Початок процесу обчислення

Таблиця 3.3 – Отримання даних від back-end(у) застосунку

Мета тесту	Отримати дані про стать та дані для графіків
Передумови	Обробка тексту закінчена
Схема виконання	Модель передає відповідь, при цьому генеруються деякі дані для графіків
Очікуваний результат	Дані надійшли на Front-end
Дійсний результат	Дані надійшли на Front-end

Таблиця 3.4 – Відсутність реакції системи

Мета тесту	Виключити ймовірність обчислень на пустих даних
Передумови	Користувач не завантажив файл
Схема виконання	Натискання кнопки «Аналізувати»
Очікуваний результат	Система не розпочинає обчислень
Дійсний результат	Система не розпочинає обчислень

Таблиця 3.5 – Завантаження замалого файлу

Мета тесту	Перевірка методу сповіщення користувача, якщо файл замалий
Передумови	Користувач завантажує замалий файл
Схема виконання	Натискання кнопки «Аналізувати»
Очікуваний результат	Система сповіщає про це користувача
Дійсний результат	Система сповіщає про це користувача

Таблиця 3.6 – Завантаження звіту аналізу

Мета тесту	Перевірка вмісту звіту та відповідність результатам
Передумови	Користувач не завантажив файл, натиснув кнопку «Аналізувати»
Схема виконання	Натискання кнопки «Завантажити звіт»
Очікуваний результат	Звіт завантажився з коректним відображенням коректних даних
Дійсний результат	Вміст звіту коректний та відповідає результатам

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання системи повинні бути встановлені Python 3.8, pip, та MongoDB. Для точності, далі буде наведено структурну схему розгортання проекту.

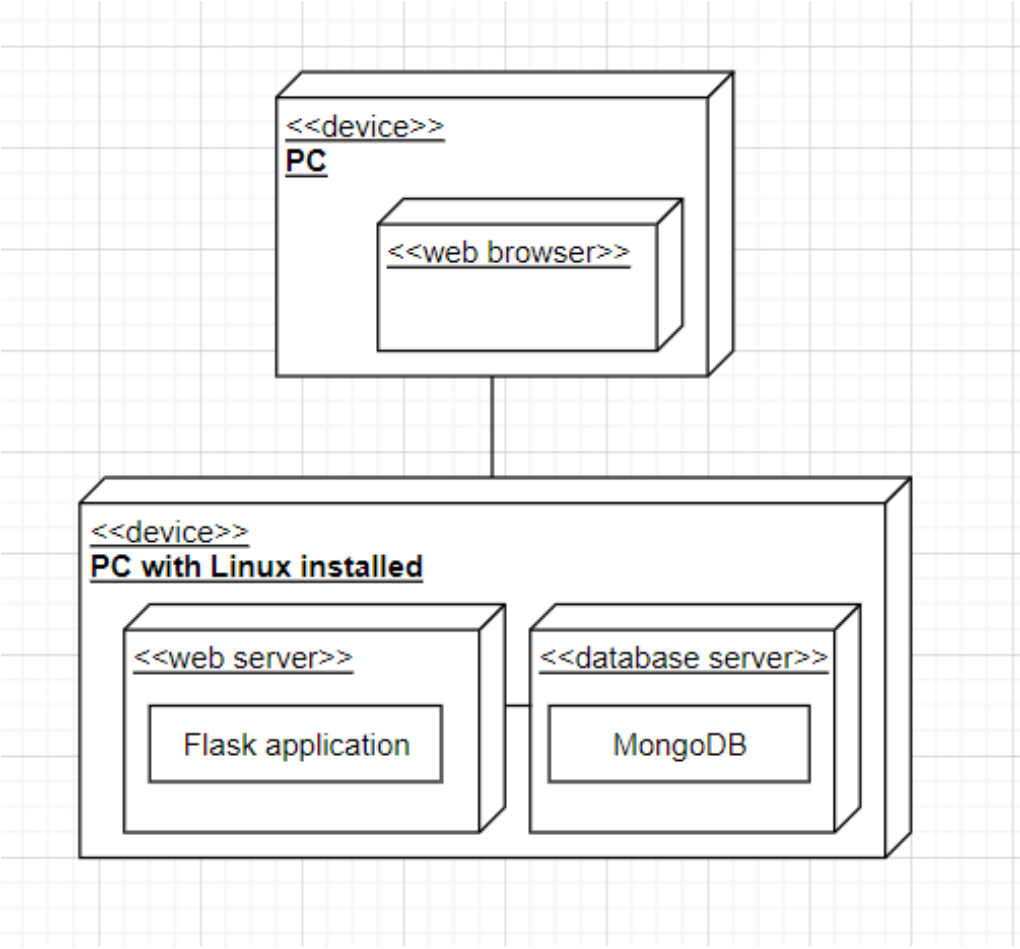


Рисунок 4.1 – Схема структурна розгортання

Проект являє собою веб-застосунок, тому для відтворення на клієнтській машині обов’язково має бути встановлений веб-браузер. У налаштування браузера повинне бути увімкнене відтворення JavaScript, оскільки цей проект використовує його для Front-end частини.

4.2 Робота з програмним забезпеченням

Робота з програмним забезпеченням передбачає наступні кроки:

– зайти на веб-сторінку за адресою розгортання (передумова – сервер запущено);

– завантажити txt файл твору;

– натиснути аналізувати;

– отримати результат у інтерфейсі;

– буде доступний посилання на файл звіту;

– за бажанням завантажити файл звіту.

4.3 Супровід програмного забезпечення

Для аналізу роботи front-end частини та передбачення помилок у JS коді можна використати вбудовані у браузер інструменти розробника. Наприклад: у Chrome натиснути F12, перейти у вкладку Console, та подивитися чи не викинув якісь помилки JS. Додатково можна зайти на вкладку Network, щоб подивитися відповіді API нейронної мережі.

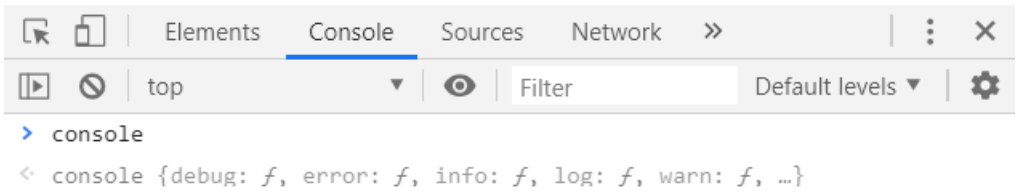


Рисунок 4.2 – Інструменти розробника у браузері Google Chrome

4.4 Висновки до розділу

Було сформовано основні вимоги для розгортання застосунку. Для розгортання підійде будь-яка машина де встановлено вище перераховані компоненти.

Для користування потрібна будь-яка машина з веб-браузером. Стисло описано як користуватись додатком та як перевірити працездатність на рівні користувача.

Додано діаграму розгортання для візуалізації взаємодії компонентів системи.

ВИСНОВКИ

Під час виконання дипломної роботи було створено програмну систему, при використанні якої користувач має змогу визначити стать автора невідомого українського тексту.

Розробка програмного продукту поділялась на декілька етапів. Перший етап містив аналіз предметної області та відомих технічних рішень задля виявлення оптимального та доцільного шляху вирішення поставленої задачі. У ході аналізу були виявлені переваги використання обраних технологій, а саме розробка нейронної мережі для аналізу вхідних даних, кодування кольорів у форматі RGB та використання аналогій. Були створені функціональні вимоги та розроблено опис сценарію взаємодії з клієнтською частиною застосунку, а також описано взаємодію клієнтської та серверної частини. Визначено мету та завдання розробки.

Наступним етапом було створення датасету для навчання моделі нейронної мережі, яке супроводжувалось розробкою словника кольорових означень. Після мінімізації втрат та оптимізації моделі, було визначено точність обчислень, яка складає близько 80 відсотків, та мінімальний обсяг вхідних даних, який дозволяє отримати коректні дані. Розроблено клієнтську частину для зручної взаємодії з користувачем.

Кінцевий застосунок відповідає всім заявленим вимогам. Мінімалістичний інтерфейс є інтуїтивно зрозумілим та зручним у використанні. Окрім того, застосунок дозволяє створити звіт з отриманих результатів у зручному форматі.

Отже, розроблена програмна система відповідає необхідним критеріям, є зручною та демонструє можливість визначення характеристик, а саме стать, автора невідомого українського тексту.

ПЕРЕЛІК ПОСИЛАНЬ

1) Nathan Moroney. Unconstrained web-based color naming experiment [Електронний ресурс] / Nathan Moroney – Режим доступу до ресурсу: http://shiftleft.com/mirrors/www.hpl.hp.com/personal/Nathan_Moroney/ei03-moroney.pdf.

2) Semantic and perceptual representations of color: evidence of a shared color-naming function. / Szeszel, Alvarado, Jameson, Sayim // Cognition Cult / Szeszel, Alvarado, Jameson, Sayim., 2005. – С. 427–486.

3) Sex-related differences in chromatic sensitivity / Rodríguez-Carmona, Sharpe, Harlow, H. Barbur // Vis Neurosci / Rodríguez-Carmona, Sharpe, Harlow, H. Barbur., 2008. – С. 433–440.

4) Introdusion to natural language processing [Електронний ресурс] – Режим доступу до ресурсу: https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32%2.

5) Lemmatization [Електронний ресурс] – Режим доступу до ресурсу: https://www.machinelearningplus.com/nlp/lemmatization-examples-python/.

6) Машинно-навчальні методи розпізнавання іменованих сутностей тексту [Електронний ресурс] – Режим доступу до ресурсу: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/126400/16-Marchenko.pdf?sequence=1.

7) Кривая ошибок [Електронний ресурс] – Режим доступу до ресурсу: http://www.machinelearning.ru/wiki/index.php?title=ROC-%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F.

8) Logistic Regression [Електронний ресурс] – Режим доступу до ресурсу: https://datascientia.blog/2017/08/14/dss-llrm/.

9) K-nearest neighbors algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

10) Відмінність RGB і CMYK кольорових моделей [Електронний ресурс] – Режим доступу до ресурсу: https://irisprintstudio.ru/info/articles/v-chem-otlichie-rgb-i-cmyk-cvetovyh-modelej/.

11) Список кольорів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%BB%D1%8C%D0%BE%D1%80%D1%96%D0%B2.

12) Keras Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://faroit.com/keras-docs/1.2.0/.

13) TensorFlow API [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/api_docs/python/.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ СТАТІ АВТОРА У ХУДОЖНІХ ТВОРАХ НА ПІДСТАВІ АНАЛІЗУ ЗГАДОК КОЛЬОРІВ

Технічне завдання

КПІ.ІП-6324.045490.03.91

“ПОГОДЖЕНО”

Керівник проєкту:

О.Д. Фіногенов

Нормоконтроль:

К.І. Ліщук

Виконавець:

В.Ю. Попова

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмна система для визначення статі автора у художніх творах на підставі аналізу згадок кольорів

Галузь застосування: Лінгвістичні дослідження, будь-яка інша галузь, яка потребує вирішення цієї проблеми

Наведене технічне завдання поширюється на розробку програмної системи для визначення авторства невідомого тесту, котра використовується для визначення статі автора художнього твору та призначена для використання у різних галузях, де потребується вирішення цієї задачі.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмної системи є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для визначення характеристик автора художнього твору та кольорове відображення певного тексту або художнього твору.

Метою розробки є можливість визначення статі автора на підставі аналізу згадок кольору у творі.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– можливість завантаження текстового файлу;

– отримання результатів щодо статі автора завантаженого тексту;

– можливість перегляду кольорового спектру завантаженого тексту;

– можливість перегляду частотних характеристик використаних назв кольорів.

4.1.2 Розробку виконати на платформі Windows 10.

4.1.3 Додаткові вимоги:

– створення словника назв кольорів та їх аналогій.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування.

Не висуваються.

4.3.3 Обслуговуючий персонал.

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на серверах та комп’ютерах з операційною системою Windows/Ubuntu/MacOS.

4.4.2 Мінімальна конфігурація технічних засобів:

– тип процесору ……. Intel Core i3;
– об‘єм ОЗП …….……... 4ГБ;
– об‘єм вільного дискового простору ……..…. 100ГБ;
– відеокарта ……NVIDIA GEFORCE з підтримкою CUDA.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем систем сімейства WIN64 (Windows 7, Windows 8 та Windows 10) або Unix.

4.5.2 Вхідні дані повинні бути представлені в наступному форматі: файл у форматі txt.

4.5.3 Результати повинні бути представлені в наступному форматі: звіт в Web-сторінка.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему.

5.3 У склад супроводжувальної документації повинні входити наступні документи.

5.3.1 Пояснювальна записка не менше ніж на 60 аркушах формату А4 (без додатків 5.3.2 - 5.3.4).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Програма та методика тестування.

5.4 Графічна частина повинна бути виконана у формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структурна варіантів використань.

5.4.2 Схема структурна бізнес-процесів.

5.4.3 Креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Таблиця 6.1 – Стадії і етапи розробки

№	Назва етапу	Строк	Звітність
1	Вивчення рекомендованої літератури	19.03.2020	
2	Аналіз існуючих методів розв’язання задачі	26.03.2020	
3	Постановка та формалізація задачі	26.03.2020	Технічне завдання
4	Аналіз вимог до програмного забезпечення	02.04.2020	Схема структурна програмного забезпечення
5	Алгоритмізація задачі	02.04.2020	
6	Моделювання програмного забезпечення	09.04.2020	
7	Обґрунтування використовуваних технічних засобів	16.04.2020	
8	Розробка архітектури програмного забезпечення	23.04.2020	Схема структурна бізнес-процесів
9	Розробка програмного забезпечення	30.04.2020	Тексти програмного забезпечення
10	Налагодження програми	07.05.2020	Програма та методика тестування
11	Виконання графічних документів	14.05.2020	Графічний матеріал проекту

Продовження таблиці 6.1

12	Оформлення пояснювальної записки	21.05.2020	Пояснювальна записка проекту
13	Подання ДП на попередній захист	06.06.2020	
14	Подання ДП рецензенту	07.06.2020	
15	Подання ДП на основний захист	19.06.2020	

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1. Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ СТАТІ АВТОРА У ХУДОЖНІХ ТВОРАХ НА ПІДСТАВІ АНАЛІЗУ ЗГАДОК КОЛЬОРІВ

Програма та методика тестування

КПІ.ІП-6324.045490.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.Д. Фіногенов

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Ю. Попова

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробувань є програмна система для визначення статі автора невідомого художнього твору, яка представлена у вигляді веб-застосунку, створеним на мові програмування Python з використанням фреймворку Flask та TensorFlow.

2 МЕТА ТЕСТУВАННЯ

Мета тестування полягає у перевірці наступних пунктів:

- функціональна коректна робота елементів веб-застосунку;
- коректна робота компонентів аналізу вхідних даних;
- відповідність сформованого звіту вихідним даним;
- забезпечення оптимального часу обробки;
- зручність використання веб-застосунку;
- відповідність системи вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування відбувається у режимі білої скриньки. Програмний продукт перевіряється на відповідність функціональним вимогам. Під час тестування даної програмної системи використовуються наступні методи тестування:

- функціональне тестування;
- компонентне тестування;
- приймальне тестування;
- тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Випробування програмного продукту відбувалось у наступних веб-браузерах:

– Google Chrome;

– Mozilla Firefox;

– Microsoft Edge.

Коректна робота веб-застосунку перевіряється наступним шляхом:

– ручне тестування, завантажуючи коректні вхідні дані у відповідне поле;

– ручне тестування, завантажуючи некоректні вхідні дані;

– ручне тестування відповідності функціональним вимогам;

– тестування компонентів;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ СТАТІ АВТОРА У ХУДОЖНІХ ТВОРАХ НА ПІДСТАВІ АНАЛІЗУ ЗГАДОК КОЛЬОРІВ

Керівництво користувача

КПІ.ІП-6324.045490.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.Д. Фіногенов

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Ю. Попова

Київ – 2020 року

ЗМІСТ

1 ОБМЕЖЕННЯ

Для завантаження вхідних даних є декілька обмежень:

– файл, що завантажується, має бути у форматі txt;

– розмір файлу повинен становити не більше 5 Мб.

2 ПОЧАТОК РОБОТИ

Перейти за адресою рішення.

Рисунок 2.1 – Адреса рішення

Далі необхідно завантажити обраний файл.

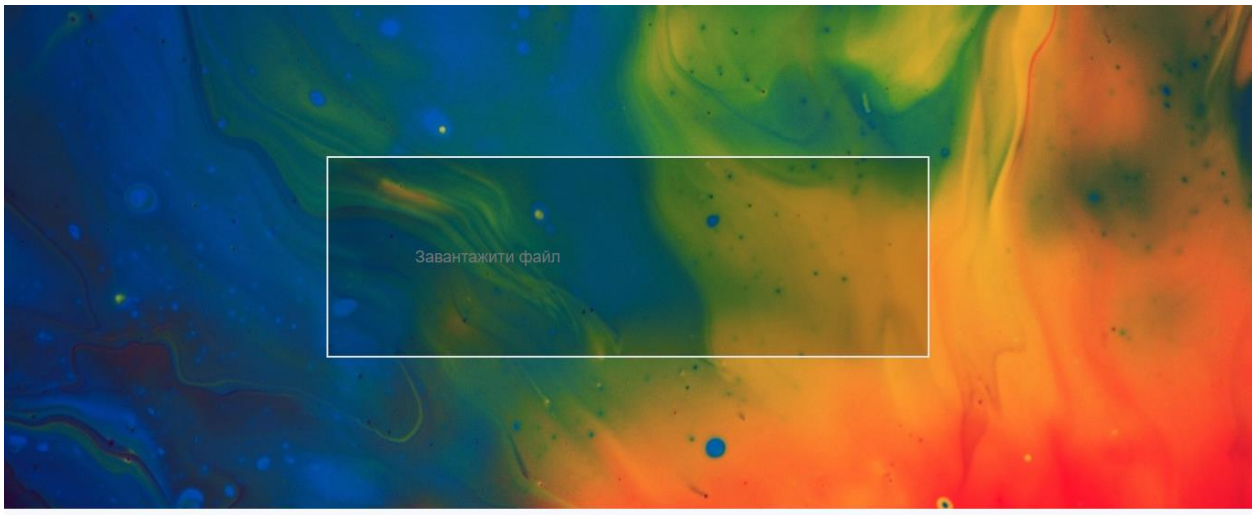


Рисунок 2.2 – Завантаження файлу користувача

Далі можна спостерігати процес роботи програми.

Рисунок 2.3 – Результати аналізу

За бажанням передбачена можливість завантаження звіту аналізу у форматі PDF та можливість поділитись результатами у соцмережах.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ СТАТІ АВТОРА У ХУДОЖНІХ ТВОРАХ НА ПІДСТАВІ АНАЛІЗУ ЗГАДОК КОЛЬОРІВ

Опис програми

КПІ.ІП-6324.045490.06.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.Д. Фіногенов

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Ю. Попова

Київ – 2020 року

Тексти програмного коду

Програмна система для визначення статі автора у художніх творах на підставі аналізу згадок кольорів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

14 арк, 12000 кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020

```
from tensorflow.python.keras import models
from tensorflow.python.keras import initializers
from tensorflow.python.keras import regularizers
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.layers import Dropout
from src.models.utils import _get_last_layer_units_and_activation

def n_model(blocks,
            filters,
            kernel_size,
            embedding_dim,
            dropout_rate,
            pool_size,
            input_shape,
            num_classes,
            num_features,
            use_pretrained_embedding=False,
            is_embedding_trainable=False,
            embedding_matrix=None):
    op_units, op_activation = _get_last_layer_units_and_activation(num_classes)
    model = models.Sequential()

    if mbedd:
        model.add(Embedding(input_dim=num_features,
                            output_dim=embedding_dim,
```

```
                input_length=input_shape[0],
                weights=[embedding_matrix],
                trainable=is_embedding_trainable))
    else:
        model.add(Embedding(input_dim=num_features,
                output_dim=embedding_dim,
                input_length=input_shape[0]))

    for _ in range(blocks-1):
        model.add(Dropout(rate=dropout_rate))
        model.add(SeparableConv1D(filters=filters,
                    kernel_size=kernel_size,
                    activation='rectified',
                    bias_initializer='random_uniform',
                    depthwise_initializer='random_uniform',
                    padding='same'))
        model.add(SeparableConv1D(filters=filters,
                    kernel_size=kernel_size,
                    activation='rectified',
                    bias_initializer='random_uniform',
                    depthwise_initializer='random_uniform',
```

```
                padding='same'))
    model.add(MaxPooling1D(pool_size=pool_size))
model.add(SeparableConv1D(filters=filters * 2,
                kernel_size=kernel_size,
                activation='relu',
                bias_initializer='random_uniform',
                depthwise_initializer='random_uniform',
                padding='same'))
model.add(SeparableConv1D(filters=filters * 2,
                kernel_size=kernel_size,
                activation='relu',
                bias_initializer='random_uniform',
                depthwise_initializer='random_uniform',
                padding='same'))
model.add(GlobalAveragePooling1D())
model.add(Dropout(rate=dropout_rate))
model.add(Dense(op_units, activation=op_activation))
return model
def load_embedding_index(filename, directory = '.'):
embeddings_index = {}
with open(os.path.join(directory, filename)) as f:
```

```
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
    return embeddings_index


def load_embedding_matrix(word_index, embedding_dim, filename, directory='.'):
    embeddings_index = load_embedding_index(filename, directory)

    embedding_matrix = np.zeros((len(word_index) + 1, embedding_dim))
    for word, i in word_index.items():
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            # words not found in embedding index will be all-zeros.
            embedding_matrix[i] = embedding_vector
    return embedding_matrix


def _get_last_layer_units_and_activation(num_classes):
    # Arguments
        num_classes: int, number of classes.
```

```
    # Returns
        units, activation values.
    """
    if num_classes == 2:
        activation = 'sigmoid'
        units = 1
    else:
        activation = 'softmax'
        units = num_classes
    return units, activation


import argparse
import time
import tensorflow as tf
import numpy as np

from src.data.preprocess import get_num_classes, sequence_vectorize
from src.models.sepCNN import sepcnn_model
FLAGS = None
def train_sequence_model(data,
                         learning_rate=1e-3,
```

```
epochs=1000,
                batch_size=128,
                blocks=2,
                filters=64,
                dropout_rate=0.2,
                embedding_dim=200,
                kernel_size=3,
                pool_size=3,
                use_pretrained_embedding=False,
                is_embedding_trainable=False,
                embeddings_folder='.',
    # Verify that validation labels are in the same range as training labels.
    num_classes = get_num_classes(train_labels)
    unexpected_labels = [v for v in val_labels if v not in range(num_classes)]
    if len(unexpected_labels):
        raise ValueError('Unexpected label values found in the validation set:'
                ' {unexpected_labels}. Please make sure that the '
                'labels in the validation set are in the same range '
                'as training labels.'.format(
                    unexpected_labels=unexpected_labels))
```

```
num_features = len(word_index) + 1
    if use_pretrained_embedding:
        embedding_matrix = load_embedding_matrix(word_index, embedding_dim,
embedding_file, embeddings_folder)
    else:
        embedding_matrix = None

    # Create model instance.
    model = sepcnn_model(blocks=blocks,
                         filters=filters,
                         kernel_size=kernel_size,
                         embedding_dim=embedding_dim,
                         dropout_rate=dropout_rate,
                         pool_size=pool_size,
                         input_shape=x_train.shape[1:],
                         num_classes=num_classes,
                         num_features=num_features,
use_pretrained_embedding=use_pretrained_embedding,
                         is_embedding_trainable=is_embedding_trainable,
                         embedding_matrix=embedding_matrix)
    # Compile model with learning parameters
```

```
    if num_classes == 2:
        loss = 'binary_crossentropy'
    else:
        loss = 'sparse_categorical_crossentropy'
    optimizer = tf.keras.optimizers.Adam(lr=learning_rate)
    model.compile(optimizer=optimizer, loss=loss, metrics=['acc'])


        callbacks = [tf.keras.callbacks.EarlyStopping(
        monitor='val_loss', patience=20)]
    # Create callback for tensorboard
    callbacks.append(tf.keras.callbacks.TensorBoard(log_dir='./tflog',
histogram_freq=1))
    # Train and validate model.
    history = model.fit(
            x_train,
            train_labels,
            epochs=epochs,
            callbacks=callbacks,
            validation_data=(x_val, val_labels),
            verbose=2,  # Logs once per epoch.
```

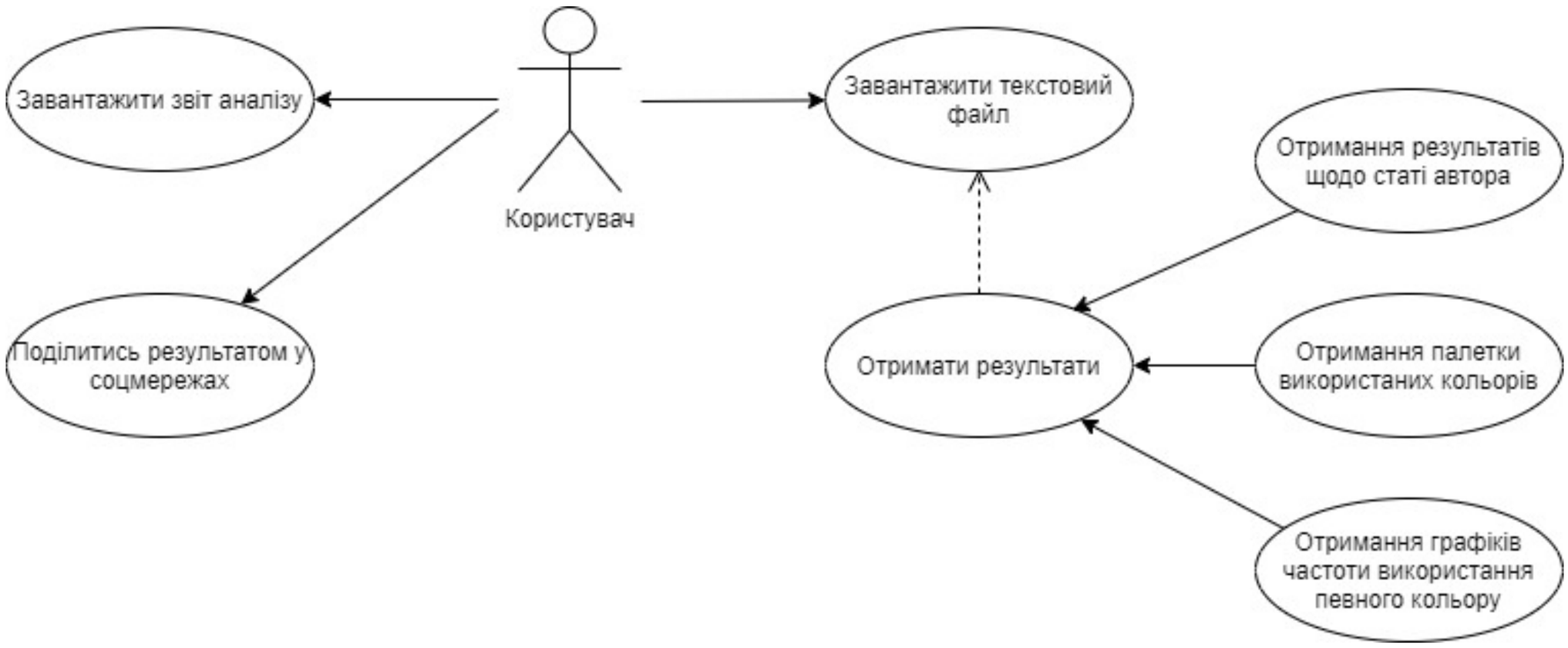
```
batch_size=batch_size)
    history = history.history
    return history['val_acc'][-1], history['val_loss'][-1]
def complete_model(weights_path=weights_filename):
    inc_model=InceptionV3(include_top=False,
                          input_shape=((3, 150, 150)))
    x = Flatten()(inc_model.output)
    x = Dense(64, activation='relu', name='dense_one')(x)
    x = Dropout(0.5, name='dropout_one')(x)
    x = Dense(64, activation='relu', name='dense_two')(x)
    x = Dropout(0.5, name='dropout_two')(x)
    top_model=Dense(1, activation='sigmoid', name='output')(x)

model = Model(input=inc_model.input, output=top_model)
    model.load_weights(weights_filename, by_name=True)
    for layer in inc_model.layers:
        layer.trainable = False
model = Sequential()
model.add(layers.Dense(10, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

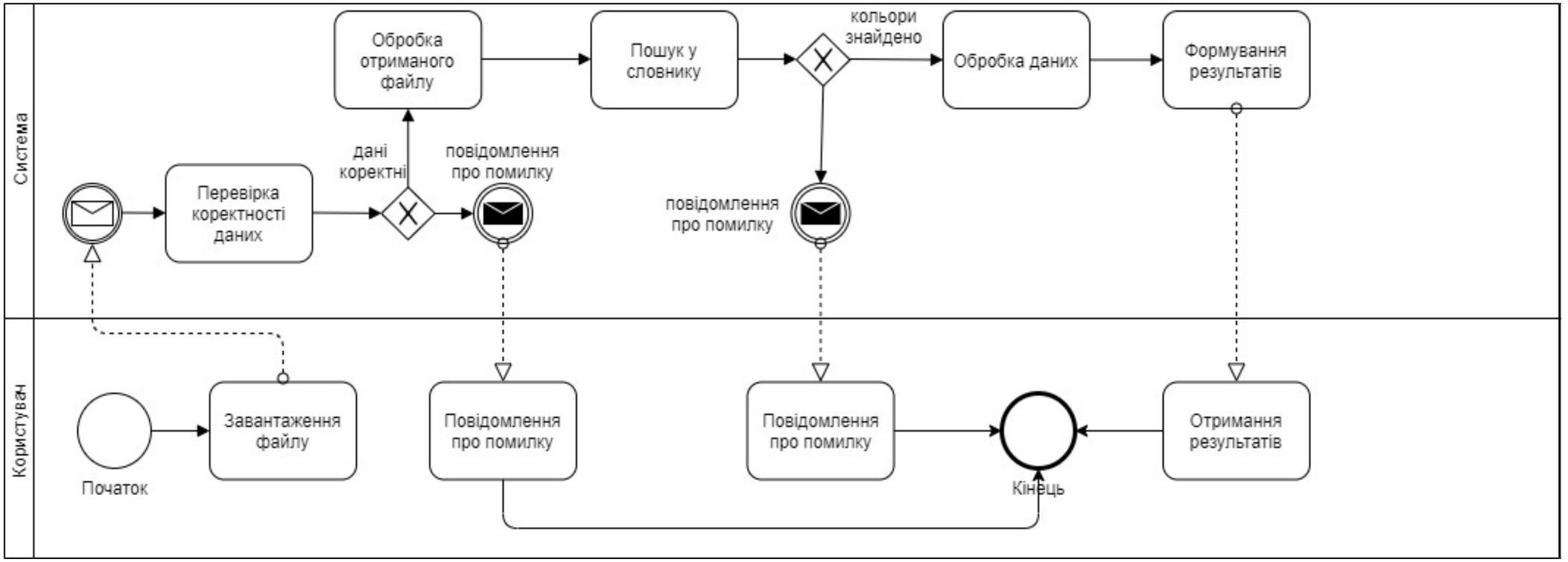
```
def create_model(num_filters, kernel_size, vocab_size, embedding_dim, maxlen):
    model = Sequential()
    model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
    model.add(layers.Conv1D(num_filters, kernel_size, activation='relu'))
    model.add(layers.GlobalMaxPooling1D())
    model.add(layers.Dense(10, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
<script>
  import { EventBus } from '../plugins/EventBus.js'
  export default {
    /*
      Defines the data used by the component
    */
    data(){
      return {
        file: ''
      }
    },
    methods: {
      /*
        Submits the file to the server
      */
      submitFile(){
        /*
            Initialize the form data
        */
        let formData = new FormData();
        /*
            Add the form data we need to submit
        */
        formData.append('textfile', this.file);
        /*
          Make the request to the POST /single-file URL
        */
```

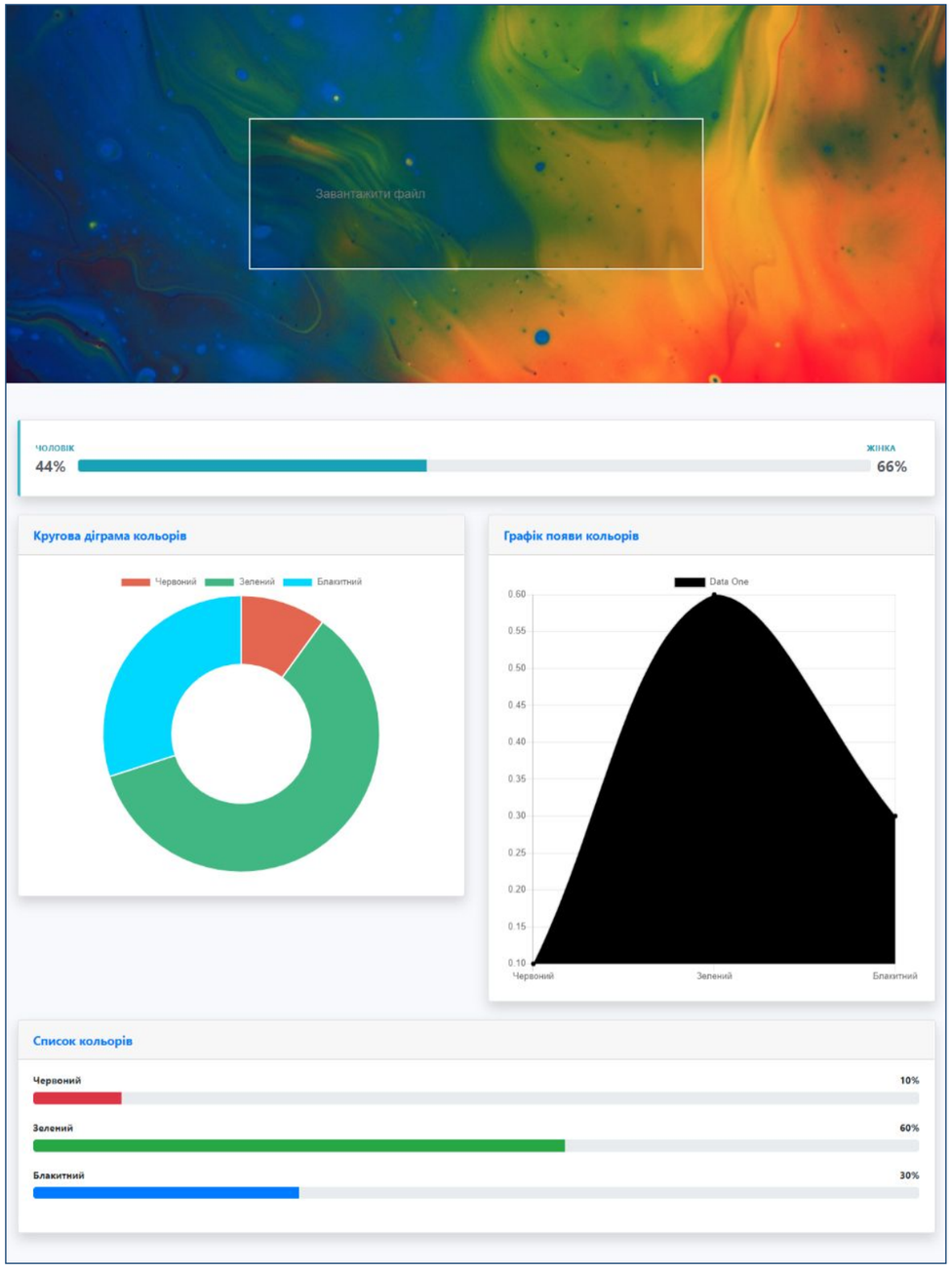
```
    /*
      Handles a change on the file upload
    */
    handleFileUpload(){
      this.file = this.$refs.file.files[0];
    }
  }
}
```

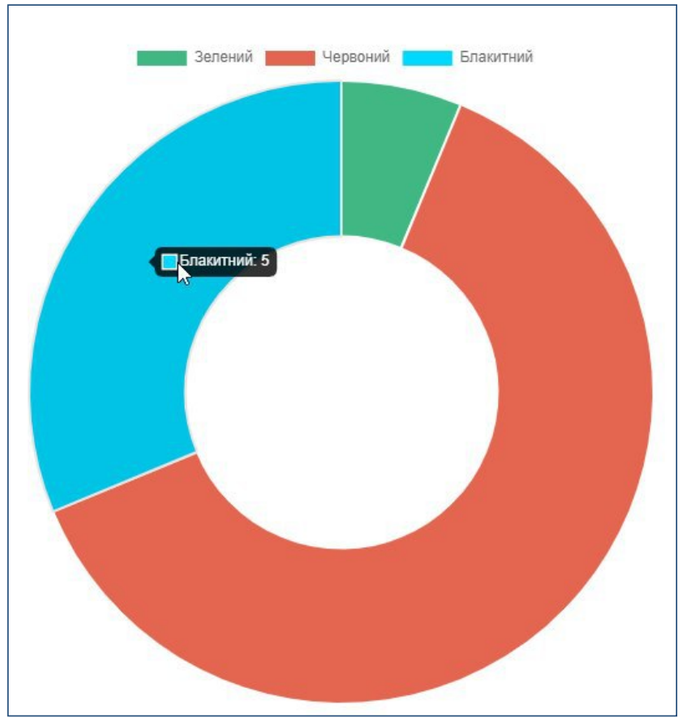


				КПІ.ІП-6324.045420.07.99.СС			
						Літера	Маса Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань		
Розробив		Попова В.Ю.				Аркуш	Аркушів
Перевірив		Фіногенов О.Д.					
Т. кон.							
Н. кон.		Ліщук К.І.			Програмна система для візначення статі автора у художніх творах на підставі аналізу згадок кольорів	КПІ ім.Ігоря СікорськогоКафедра АСОІУгр. ІП-63	
Затвердив		Фіногенов О.Д.					



					КПІ.ІП-6324.045420.08.99.СС
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна бізнес-процесів
Розробив		Попова В.Ю.			Літера: Маса: Масштаб:
Перевірив		Фіногенов О.Д.			Аркуш: Аркушів:
Т. кон.					Програмна система для візначення статі автора у художніх творах на підставі аналізу згадок кольорів
Н. кон.		Ліщук К.І.			КПІ ім.Ігоря СікорськогоКафедра АСОІУгр. ІП-63
Затвердив		Фіногенов О.Д.			





				КПІ.ІП-6324.045420.09.99.КЕ		
Зм.	Арк.	№ документа	Підпис	Дата	Літера	Маса Масштаб
Розробив		Попова В.Ю.				
Перевірив		Фіногенов О.Д.			Аркуш	Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.				
Затвердив		Фіногенов О.Д.				

Креслення вигляду екранних форм

Програмна система для визначення статі автора у художніх творах на підставі аналізу згадок кольорів

КПІ ім.Ігоря СікорськогоКафедра АСОІУгр. ІП-63